University of Southern Queensland

Faculty of Engineering & Surveying

# Development of a Remote Laboratory System for External University Students

A dissertation submitted by

A. Birkbeck

in fulfilment of the requirements of

**ENG4112 Research Project**

towards the degree of

**Bachelor of Engineering (Computer Systems)**

Submitted: October, 2007

# Abstract

This research project develops Remote Laboratory Software to allow students to perform laboratory experiments from off campus. The software allows students to externally use a Remote Laboratory, which is an experiment laboratory that is specially setup so students can operate it as if they were in the laboratory itself.

Three times a year external students travel to the University of Southern Queensland for their practical experiments at residential school. As external students often live, and hence have families and work commitments, in different cities or other states it is difficult for them to travel to the university campus at the required time. It was therefore both practical and useful to develop a Remote Laboratory System that students use to do selected experiments from home, reducing the time they are required to be on campus.

Currently there are several Remote Laboratories in Australia, as a number of these systems were experiencing difficulties, a new system designed specifically for the University of Southern Queensland's needs was a viable alternative.

The project enables students to complete selected remote laboratory experiments set up at the University of Southern Queensland. They do this by using any computer with Internet access throughout the world to perform the experiments, discuss outcomes with other students, and be evaluated by practical course examiners.

University of Southern Queensland
Faculty of Engineering and Surveying

**ENG4111/2 *Research Project***

## Limitations of Use

The Council of the University of Southern Queensland, its Faculty of Engineering and Surveying, and the staff of the University of Southern Queensland, do not accept any responsibility for the truth, accuracy or completeness of material contained within or associated with this dissertation.

Persons using all or any part of this material do so at their own risk, and not at the risk of the Council of the University of Southern Queensland, its Faculty of Engineering and Surveying or the staff of the University of Southern Queensland.

This dissertation reports an educational exercise and has no purpose or validity beyond this exercise. The sole purpose of the course pair entitled "Research Project" is to contribute to the overall education within the student's chosen degree program. This document, the associated hardware, software, drawings, and other material set out in the associated appendices should not be used for any other purpose: if they are so used, it is entirely at the risk of the user.

**Prof F Bullen**
Dean
Faculty of Engineering and Surveying

# Certification of Dissertation

I certify that the ideas, designs and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.
I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

A. Birkbeck

0050007144

---

Signature

---

Date

# Acknowledgments

Thanks to Steve Murray at The University of Technology, Sydney, Lachlan Webb at University of Wollongong, Clive Ferguson at Deakin University, Euan Lindsay from Curtin University, and James Trevelyan from University of Western Australia for sharing their knowledge and information about the remote laboratories that they had dealings with.

Thanks to USQ lectures for providing me with practical course books which provided information about what experiments are currently being run at USQ.

Thanks to John Leis for being my voice, arms, and legs in Toowoomba for information gathering and for the numerous consultations and assistance he offered throughout the year. Lastly thank you to Merridee McKay who has been a guinea pig for my ideas, and designs, and had the distasteful pleasure of proof reading my first draft.

Thank you!

A. Birkbeck

*University of Southern Queensland*
*October 2007*

# Contents

# List of Figures

# List of Tables

# Glossary

**Client computer** is the computer that is used by the remote user to access the laboratory.

**Host Computer** is a program that is to control the laboratory equipment.

**RLS** Remote Laboratory Software

**Web Server** is a computer that hosts the software that the remote user will interact with.

**USQ** is the University of Southern Queensland.

**VPN** stands for Virtual Private Network and is a private data network that utilizes a public telecommunication infrastructure.

# Chapter 1

# Introduction

## 1.1 What is a Remote Laboratory?

For the purposes of this research project a Remote Laboratory is an experiment laboratory that is specially setup so that laboratory technicians can operate it as if they were in the laboratory itself. Using a computer the technician interacts with a Remote Laboratory Program that sends information via the Internet to the Remote Laboratory where the instructions are executed by robotics in the previously set up laboratory. This improves the safety and convenience of performing an experiment by allowing it to be performed from the next room or even across the world.

## 1.2 Objectives

This aim of this research project is to develop the necessary software to implement a remote laboratory that would be suitable for performing several practical experiments which are currently conducted by university students. The remote laboratory software will operate on computer systems that are common to both universities and the wider community, so that the system could be used by a home user.

## 1.3 Implementation

External students are required to attend residential school at the University of Southern Queensland numerous times throughout there studies where laboratory experiments are performed for various subjects. Although attending residential school exposes students to a wide variety of equipment, methods, and experiences that are essential for their education there are many possible experiments offered in the practical courses at the University of Southern Queensland that have the potential be run as a remote laboratory experiment. By implementing selected experiments as remote laboratories students will be able to obtain the understanding that they would normally gain from attending residential school without attending the university, which would save them both time and money.

## 1.4 Consequential Effects

While developing the remote laboratory software many aspects of sustainability, safety and ethics were considered.

- The set up required for the remote laboratory system will not undermine the development and environmental needs of future generations (*Towards Sustainable Engineering Practice: Engineering Frameworks for Sustainability* 1997) as the system uses pre-existing communication lines. The only exception would be if the system is desired by a person in an undeveloped area (i.e. isolated rural), in which case the initial establishment of communication lines may influence developmental and environmental needs, but the installation of these lines would be negotiated with the local council who take all precautions to make developments while still working towards a sustainable future.

- The development of this project does not influence environmental protection issues (*Towards Sustainable Engineering Practice: Engineering Frameworks for Sustainability* 1997) as it is software that is being developed, and as such has no immediate or long term effects on the environment. Once the system is running there will be a minute influence in power consumption by the running of computers to operate the software, but not enough to notably effect the environment. Also, as previously mentioned, if additional lines are required environments may be varied with the development of these lines.

  Alternatively there are positive environmental side effects from the development of this software. As the system is accessed remotely people no longer need to commute to the laboratory site, therefore would create less vehicle pollution.

- Possible global environmental impacts (*Towards Sustainable Engineering Practice: Engineering Frameworks for Sustainability* 1997) are very minimal. Long term electricity sources will be required, so for optimal environmental preservation the ideal system is a renewable energy source (i.e. hydro, wind or solar power). Should the system become vastly used then high capacity communication lines will be required. This will not impact the environment or future development as existing phone lines will simply be replace by fibre-optic ones.

- The only aspect of this projects implementation that could cause environmental degradation (*Towards Sustainable Engineering Practice: Engineering Frameworks for Sustainability* 1997) is the long-term side effects of energy consumption, which is currently being researched by scientists around the world.

- The participation of any concerned citizens (*Towards Sustainable Engineering Practice: Engineering Frameworks for Sustainability* 1997) would be necessary with the implementation of this project only in cases were additional communication lines are required. In these cases any concerned parties (i.e. land owners, conservationists, and developers) would be involved in discussion with the council before lines were constructed.

- Pollution will not be created by the implementation of this project; it stands to possibly have a positive effect on pollution levels (as previously mentioned with decreased vehicle pollution); hence will acquire no additional costs.

- The remote laboratory system will be equally accessible to all groups of people whether majority and minority (*Towards Sustainable Engineering Practice: Engineering Frameworks for Sustainability* 1997); the only requirement is a computer and access to the Internet. People in a low socioeconomic group who cannot purchase these requirements would access them elsewhere (local library or university), which they would already be necessary for other aspects of their university studies.

  Possible effects on employment are: there will be less demand of Toowoomba resources (accommodation, tourism, and retailers) as less people would be commuting there yearly. To offset that more employment opportunities would exist in the construction, installation, and maintenance of the system. While employment opportunities still exist the necessary qualifications vary significantly from the previous employment market to the new job fields. Further education or training may be required by job seekers but they would be rewarded by entering a higher paying job field.

- The remote laboratory software will not be used in under developed countries (*Towards Sustainable Engineering Practice: Engineering Frameworks for Sustainability* 1997) as the software is being developed for university laboratory experiment purposes. There will be no effect for countries accessing the university, as no labour is increased or decreased in that country if students travel to the USQ campus or access the system remotely.

- This project development could be used for global sustainability and international understanding (*Towards Sustainable Engineering Practice: Engineering Frameworks for Sustainability* 1997) as it creates opportunities for communication and interaction between globally diverse students.

## 1.5 Methodology

1. *Research current remote laboratories to discover the basic structure of how a remote laboratory works.*

   This research will be performed by reading reference material and by contacting the administrators of several remote laboratories located within Australia and requesting any documentation or manuals explaining how the software operates. This may also include visiting current sites if this can be arranged.

2. *Investigate how other programmers have developed their software.*

   If possible gain access to operating remote laboratories, source code used to operate the laboratory equipment or, alternatively, download a copy of the programs used, and explore their operation and features.

3. *Identify flaws associated with currently used methods.*

   Through current program exploration an understanding of how the program operates will be gained. Then the programs strengths and weaknesses can be identified.

4. *Research the software and hardware available at USQ and other potential consumers.*

   Through investigation of the current market for remote laboratories it is possible to determine the likely clients and through contact with the IT department of the clients, it is possible to determine how their system is currently running.

5. *Identify which platform the system will run on.*

   A platform, in which the system will operate on, will be decided based on the above points mentioned; Current software, flaws in current software, and clients' software and hardware.

6. *Determine any additional hardware required for the remote laboratory.*

   After identifying what hardware the clients network uses and what platform the system will run on additional hardware requirements to run the remote laboratory can be determined.

7. *Evaluate hardware requirements and recommend a vendor.*
   Tabulate the advantages and disadvantages of each of the hardware manufactures and weigh these options up will determine the best hardware manufacturer(s) for the tasks.

8. *Research alternative methods for the tasks required to implement the remote laboratory software.*
   By consulting current journals and utilising current technology a list of alternative to each task can be created.

9. *Evaluate the alternative methods of performing those tasks.*
   Each alternative method will tabulated showing the positive and negative effects of each method to determine the best alternative solution to the current method of performing a task.

10. *Assess programming languages to determine which best implements the remote laboratory, and can operate on the hardware and software used by the consumers.*
    Each alternative method can be tabulated with higher weights given to important features, and lower weights given to less important features so to determine the best alternative solution to the current method of performing a task.

11. *Develop a prototype of the remote laboratory software.*
    Code the software for the remote laboratory by completing the tasks required using the methods decided on.

12. *Extensively test the system to determine flaws.*
    Setup several test experiments and perform them as they would be performed by a remote user. Perform operations that wouldn't normally occur in an effort to try to break the system. If a flaw is found correct it.

13. *Gain user feedback on the operation of the system.*
    Setup an experiment and allow potential client users to operate the experiment and get them to complete a survey on what they thought of the system. Determine ease of use, flexibility, learning experience, and overall satisfaction.

14. *Implement changes to correct any flaws identified from testing and feedback.* Redesign any aspects that experienced problems or that the user wasnt happy with based on the information in the surveys.

## 1.6 Risk Assessment

### 1.6.1 During execution of the project

- In the designing stage of this project the first risk could be incorrectly programming the equipment. The exposure to this risk is very rare (only once during the programming stage) and the likelihood of it occurring is slight. If programmed incorrectly the consequences would be damage incurred to equipment. This risk is avoidable with careful attention being given while programming. In the event of occurrence new equipment would need to be purchased.

- During the setup of a laboratory experiment a risk of electrocution exists. The likelihood is significant but the exposure will vary depending on how many different experiments are used. Consequences may vary from minor injury to possible death. The electrical power source hazard is unavoidable as it is necessary for the experiment but the risk can be drastically minimised by checking all equipment for faults before using, turning the power source off while connecting, and confirming circuit is wired correctly. Also in case of emergencies a fire extinguisher should be easily accessible.

  Minor and major damage could also occur to equipment if this incident occurred and the same prevention steps as above would need to be taken.

- Once the equipment is connected and running there is the risk of a short circuit in an experiment that involves an electrical circuit. The likelihood of this occurring is slight and the exposure will vary depending on how many different experiments and what sorts of experiments are used. As the laboratory will be being accessed remotely no human injury will arise in the occurrence of a short circuit, however there is a chance of minor through to major equipment damage occurring. To minimise this risk experiment circuits will need to be designed so that no short circuits occur mistakenly, and circuit breakers can be integrated into each of the experiments.

- When software system is operating on the Internet or accessible to the public there exists a risk that the system will be hacked into and manipulated in ways in which the software was not designed to operated. The likelihood of such an event occurring varies depending on the nature of the users who operate the system and number of people who access the system. The risk can be reduced by locating the software on a secure server and programming in several safeguards that stop the system from operating should an attack be detected. The consequences from the system being hacked vary from an unauthorised access to damage to equipment or the Remote Laboratory Software.

- While users are operating the software they will be working on a computer frequently for numerous hours (times will vary depending on the experiment requirements) so they could stand a chance of getting eye fatigue from looking at the monitor, or repetitive strain injury (RSI) from using the mouse, keyboard or sitting at a desk. The likelihood of either of these occurring is very slight but precautionary measures can be taken to minimise the risk even further. Practicals could be designed to only go for short periods of time allowing the user to have a break between experiments. Users could also be prompted with designed short movement activities (away from the computer desk) to be completed at appropriate intervals during the experiment process to give them a break.

### 1.6.2 Beyond the completion of the project

- If the remote laboratory software is destroyed at the completion of this project no adverse effects will occur to the environment or society.

- If the project is taken onboard and used by the university all the previously mentioned risks will still be relevant and must be controlled with the recommended procedures.

## 1.7 Resources Required

The development Remote Laboratory Software requires access to a computer which will be used to code and test the software. Development also requires the appropriate programming language tools, such as compilers and linkers in order to implement a solution. During the development stage a laboratory interface device will also be needed to test the software and, if required, a program that is used to run the website. A feedback device is also needed to ensure that the user can understand what has occurred in the remote laboratory.

Currently all required resources were already possessed from previous unrelated programming projects, except the laboratory interface device which can be purchased from a selected computer specialist parts supplier.

## 1.8 Background Research

Currently there are many Remote Laboratories in Australia, several of which are used for educational purposes. Table 1.1 shows several Universities in Australia that have remote laboratories and how they are operating.

It was determined that the majority of these laboratories operated in a method that, for the purpose of this project, will called the "interface and host method". This method operates by:

- The client computer interacts with an interface programs which allows the users to change parts of the experiment. This is often a website or customised software that is developed specifically for a particular experiment.
- The interface program then communicates these changes with a host program by using a log file which then interprets these instructions and operates the machinery or circuitry to perform the requested actions.
- The feedback of the actions performed in the laboratory are then streamed back to the interface program which displays this to the user.

Predominately these remote laboratories were written in Java for the web interface, and are written in portable programming languages such as C and C++ for the host program. They generally provide video and audio feedback to the user so that they can easily understand what is occurring.

Many remote laboratories studied operate like the remote laboratory at the University of Technology, Sydney which utilises using a web server. This server hosts the website that students singularly use to interface with the remote laboratory, and communicates with the host computer, which then interacts with the laboratory equipment allowing students to control the operation of an experiment (Murray & Lasky 2007). The remote laboratory at the University of Technology, Sydney also provides additional features not offered by other systems as students are able to upload code directly to dedicated test circuit boards and see the results of their programs via live video feed (Murray & Lasky 2007).

The Telelabs Project, created at the University of Western Australia, is a remote laboratory that operates differently to the others. Students are permitted to connect to the system to perform experiments as a group, sharing control over the actions that take place. This laboratory operates by a series of software packages that are installed and run on the client computer to control the experiment (*The Telelabs Project* 2007).

| Site Name | Contact | Currently Operational | Interface | Feedback | Author | Programming Languages Used |
|---|---|---|---|---|---|---|
| Curtin University of Technology | Euan Lindsay | Redevelopment | | Video and Audio Feed | Euan Lindsay | MatLab and TightVNC |
| Deakin University | Clive Ferguson | No. Software not working | Web Browser | Video Feed | | |
| University of Technology, Sydney | Steve Murray | Yes | Web Browser | Video and Audio Feed | Undergrads and External programmers | C and Unix Shell |
| University of Wollongong | Lachlan Webb | Redevelopment | Web Browser | MPEG2, Images | Dr. Phil Cuifo, David Atkinson and Stein Krav | C and Perl CGI |
| University of Western Australia | | | Download Software and Web Browser | | | |

Table 1.1: Current Remote Laboratories

## 1.9 Practical Courses

Currently there are several practical courses running in the engineering department of the University of Southern Queensland. Tables 1.2 to 1.5 show a list of several practice courses offered by USQ, the experiments that are performed in that course, and whether the experiment can be performed remotely.

The information interpreted in these tables was sourced from the practical books of; Civil Materials Practice, Computer Systems Engineering Practice, Field Practice, Mechatronic Practice 1, Professional Practice 1, Professional Practice 2, Electrical and Electronic Practice A, Electrical and Electronic Practice B, Electrical and Electronic Practice C, Electrical and Electronic Practice D, Soil and Water Engineering Practice 1, and Soil and Water Engineering Practice 2.

Experiments have been classified as not able to be performed remotely if:

- The experiment only required watching a seminar or video, which would be best achieved with the students watching a live video feed.
- The experiment would best work with a remote desktop connection. In these situations this experience can be obtained by allowing students to connect directly to university computers in which the software package is installed.
- The activities are too difficult to perform remotely because of the actions required; I.E. extensive equipment set up, configuring time, and possibilities of voiding warranty. For example, many experiments involve the use of a Cathode Ray Oscilloscope. To operate an oscilloscope remotely would require a large number of connections interfacing with the oscilloscope and these connections would possibly void the devices warranty. A better alternative is to operate a PC-based oscilloscope via a remote desktop connection.
- The understanding gained by the experiment is lost when performed remotely.

| Practical Course | Experiments | Performed Remotely? | Req. | Feedback Required |
|---|---|---|---|---|
| Civil Materials Practice | | | | |
| | Aggregate | No | | |
| | Concrete | No | | |
| | Timber | No | | |
| | Bitumen, | | | |
| | Asphalt, | | | |
| | and Skid | | | |
| | Resistance | No | | |
| | Soil | No | | |
| | Traffic Study | No | | |
| | Road Maintenance | | | |
| | Needs Assessment | | | |
| | Study | No | | |
| Computer Systems Engineering Practice | | | | |
| | 1.1 to 2.6 | No | | |
| Field Practice | | | | |
| | Oral | | | |
| | Presentation | No | | |
| | Report | No | | |
| Mechatronic Practice 1 | | | | |
| | Parts 1 to 6 | No | | |
| Professional Practice 1 | None | | | |
| Professional Practice 2 | None | | | |

Table 1.2: Engineering Practical Courses Offered by USQ

| Practical Course | Experiments | Performed Remotely? | Req. | Feedback Required |
|---|---|---|---|---|
| Electrical and Electronic Practice A | 1 | No | | |
| | 2, 3 & 4 | Yes | 4 digital switches | Visual |
| | 5 & 6 | No | | |
| | 7 & 8 | Yes | 2 digital switches | Visual |
| | 9 | No | | |
| | Act. 5.2 to 5.6 | No | | |
| | A1 | No | | |
| | A2 & A3 | Yes | 1 digital & 2 analog switches | Visual |
| | B1 & B2 | Yes | 2 digital & 1 analog switches | Visual |
| | C1 | Yes | 1 digital | Visual |
| | C2 | No | | |
| | C3 | Yes | 3 digital & 1 analog switches | Visual |
| | D1 | Yes | 1 digital | Visual |
| | D2 | Yes | 3 digital & 4 analog switches | Visual |

Table 1.3: Engineering Practical Courses Offered by USQ

| Practical Course | Experiments | Performed Remotely? | Req. | Feedback Required |
|---|---|---|---|---|
| Electrical and Electronic Practice B | | | | |
| | 1 to 14 | No | | |
| Electrical and Electronic Practice C | | | | |
| | 1.1 to 2.6 | No | | |
| | 3.1 | No | | |
| | 3.9 | Yes | 2 digital & 2 analog switches | Visual |
| Electrical and Electronic Practice D | | | | |
| | Act. 1.1 to 4.1 | No | | |
| | Act. 5.1 | Yes | 3 digital & 1 analog switches | Visual |
| | Act. 5.2 | Yes | 3 digital & 1 analog switches | Visual |
| | Act. 5.3 | Yes | 2 digital & 2 analog switches | Visual |
| | Act. 5.4 | No | | |
| | Act. 5.5 | No | | |
| | Act. 5.6 | Yes | 3 digital & 2 analog switches | Visual |

Table 1.4: Engineering Practical Courses Offered by USQ

| Practical Course | Experiments | Can be performed RL | RL Requirements | Feedback Required |
|---|---|---|---|---|
| Soil and Water Engineering Practice 1 | | | | |
| | 2.1 | No | | |
| | 3.1 | No | | |
| | 3.2 | Yes | 2 analog switches | Visual |
| | 4.1 | Yes | 1 analog switch | Visual |
| | 5.1 | Yes | 2 analog switches | Visual |
| | 5.3 | Yes | 3 analog switches | Visual |
| | 6.1 | Yes | 1 analog switch | Visual |
| | 6.2 | Yes | 1 analog switch | Visual |
| | 7.1 | Yes | 1 analog switch | Visual |
| | 9.1 | Yes | 1 digital & 2 analog switches | Visual |
| | 10 | No | | |
| Soil and Water Engineering Practice 2 | | | | |
| | 1 to 8 | No | | |

Table 1.5: Engineering Practical Courses Offered by USQ

## 1.10 Proof of Completion

In order for the Remote Laboratory to become an integral part of the practical courses the experiments which are performed remotely need to provide adequate proof of student completion to the course examiner. Currently students attend residential school and perform experiments in the presence of a USQ staff member who signed off on an experiment once they have witnessed the student performing it. With the experiment being performed remotely in a laboratory students are not being overseen constantly and, as such, the need arises for evidence that the student has completed the experiment. Evidence of students' successful completion of experiments can be gathered in several different ways.

As with the current method a USQ staff member could be present in the remote laboratory to ensure that all current experiments are being completed correctly. Students would log into the system and perform the experiments and the supervising staff would sign them off once the experiment was completed correctly. This staff member would also need to be able to communicate with the student completing the experiment so that they could inform them of anything that was being performed incorrectly.

Alternatively a software solution could be implemented in which students are required to take screen shots at important milestones in the experiment. These screen shots are to be submitted along with any calculations to the examiner as proof that the student has completed the experiment as required by the practice course.

The first method requires a staff member to be constantly present in the remote laboratory to ensure that each practice course is completed correctly. In doing so the hours that a remote laboratory experiment can be performed is reduced. The latter alternative allows students to interact with the experiment and to save a screen shot at their leisure. It doesn't restrict the hours that the experiment can be performed in and is therefore the chosen alternative for this project.

# Chapter 2

# Interface and Host Program Configuration Alternatives

Using the information gained in the background research it was established that the "interface and host" method was the best approach for the purposes of this project. This interface and host method involves two programs that operate independently to complete the remote laboratory.

The host program is primarily responsible for periodically checking the log (created by the interface program), verifying the instructions, and implementing the requested actions. The host program would also be responsible for preparing the feedback of the experiment so that the website may portray the feedback to the user.

The interface program is responsible for interaction with the users. Users access this program which first authenticates them, then prompts the users to select an experiment. Once an experiment is entered the users are given the option to manipulate the experiment as they wish, within the bounds of the experiment. Once a user has implemented a change the instructions are saved to a log file. The Interface program can be achieved by two methods; website and client program.

The cost of implementing the interface and host program based solution depends on the programming language that the site is to be written in, and whether it is being hosted locally at the university, or remotely by an external provider.

## 2.1 Interface Program Connections

In the interface and host method, users are required to interact with the interface program in order to be able to access the remote laboratory. There are several ways in which the user can interact with the interface program.

### 2.1.1 Web Browser

This alternative allows the users of the remote laboratory to use any recent web browser to access the remote laboratory. This has the advantage of not requiring specialised software to be installed in order to access the remote laboratory which is beneficial for those users who are trying to access the laboratory from a restricted network where software cannot be installed.

### 2.1.2 Remote Desktop Connection Based

For a remote desktop connection based system a Microsoft terminal server could be installed at the university. This would allow external and internal students to connect to the university using client programs such as Microsoft Remote Desktop or similar and allow them to perform experiments without attending the university. In this situation only one program would need to be coded to allow the manipulation of the laboratory equipment. This method is advantageous because it also allows experiments such as the LaTeX and the Protel introduction courses to be conducted as a remote laboratory. A Remote Desktop program comes standard with Windows XP, and Vista and is available on Linux and Mac from such providers a RealVNC (*RealVNC* 2007).

### 2.1.3 VPN Based

The user could use a VPN connection to connect to the university network. This would enable them to access the interface program without occupying any university computers. The VPN based method provides greater network security then using a remote desktop connection as no additional external ports would need to be opened if a VPN pass through router was used. This method however would also allow potential hackers access to the university network should they be able to connect via VPN. VPN is freely available on Mac, Windows, and Linux platforms (*Using a Linux L2TP/IPsec VPN server with Mac OS X* 2007).

### 2.1.4 HTTP Tunneling

HTTP Tunneling works by getting the client to encapsulate data within a HTTP header. This is then received by the server which strips the HTTP encapsulation header from the received packet and forwards it to the requested destination. HTTP Tunneling can be used to encapsulate TCP and UDP packets (*Data Driven Attacks Using Http Tunneling* 2007). Network administrators will go to severe lengths to ensure that users cannot get onto the Internet using HTTP Tunnels. This is because allowing users HTTP Tunnel in a network would clear the way for any program to access the Internet through the port 80 (*Data Driven Attacks Using Http Tunneling* 2007). The remote laboratory can be setup to run using a HTTP tunnel but it does create a security risk at the university by making it more susceptible it external attacks and as such should not be used. Currently HTTP tunneling is only available in the Windows environment (*Http- Tunnel* 2007) and as such limits the different platforms that this solution may be implemented on.

### 2.1.5 Conclusion

The Web Browser approach provides clear advantages over VPN based and HTTP Tunneling method in security. The VPN, Remote Desktop, and HTTP Tunneling methods provide direct access to the university computer network creating a greater security threat on their operations whereas the Web Browser based solution only allows users to connect to the interface. The Web Browser method is more portable then The Client Program method because it can be run from any computer with Internet access. The Web Browser method will require that a website be written to make use of this method. Strict security policies enforced on the university network only permit the Web browser method to be utilised.

## 2.2 Interface Program

The interface program is the program that the user interacts with in order to make changes in the remote laboratory. The interface program may be created in two different methods.

### 2.2.1 Website

The website based solution could be configured using or not using the host computer as the web server. If the host computer is the web server then the system gains the benefits of less computers required to run the software and thus less cost. If the host computer is not the web server then the system benefits by not requiring a web server to be located on campus. The web server can then be hosted by a specific provider which reduces the computer requirements at the university in order to run the system.

### 2.2.2 Client Program Based

The client program based method requires that the user download and install programs onto their computer. These programs work with the users Internet to establish a connection to the university's remote laboratories where the client programs interact with a host programs. This can cause problems on secure networks were users are not allowed to install programs, such as work places, and Internet Cafe's.

### 2.2.3 Conclusion

| Method | Cost | Portability | Security | Ability to complete task |
|---|---|---|---|---|
| Website | ⋆⋆⋆⋆ | ⋆⋆⋆⋆⋆ | ⋆⋆⋆⋆ | ⋆⋆⋆⋆⋆ |
| Client Program | ⋆⋆⋆⋆ | ⋆⋆⋆ | ⋆⋆⋆ | ⋆⋆⋆⋆ |

Table 2.1: System Configuration Methods Summary

University network security should be given the highest priority to avoid potential security breaches. The ability for the system to complete the task should be the second highest priority, with cost and portability being of equal value as third most important. With these priorities in mind the most secure solution is the Website, which also scores the highest in its ability to achieve the task. This system is portable to PC, Mac, and LINUX and is very cost effective, depending on the programming language selected.

# Chapter 3

# Hardware Requirement Alternatives

## 3.1 Laboratory Equipment Interface Method

The system will need to be able to interface with the equipment in the laboratory in such a way that experiment parameters may be changed by the user.

### 3.1.1 Analog Switches

An analog switch provides the user with the ability to adjust the circuit to a particular value. A digital switch allows to user to turn on and off a device such as a motor, whereas a analog switch would provide the user with the ability to change how fast the motor would operate. An analog switch could be manufactured using a digital to analog converter. This would generate a voltage that varies depending on the user input. This voltage could then be used as in input into a servomechanism to allow the user to accurately control many different applications such as motor speed, resistor values, or room temperatures.

### 3.1.2 Robotic Arm

A robotic arm provides a more hands on approach to the experiments. The user would be in direct control of the operation of a robotic arm in which they can make changes as if they were doing it in person.

When using a robotic arm to control an experiment a new laboratory experiment can be setup easily simply placing all the equipment within the reach of the robotic are which the user has control of. The user can then more the equipment into position to yield the desired result from the experiment.

In order to accurately control the operation of the robotic arm the user must receive real-time feedback. Any delay in the feedback system would cause the system to be difficult to use. The robotic arm would make the experiment more prone to short circuits. The arm would need to be insulated so as to not short circuit the experiment. Secondly, this increase the chances that the user will connect a conductor incorrectly causing damage to the circuit and in more extreme circumstances, the building wiring.

There are many robotic arms available today from many different manufacturers. The robotic arms can be controlled via joystick, mouse, keyboard, script, and via coordinates making the systems very versatile to control. Robotic arms vary in price greatly depending on quality of the device and the movement that the device provides (*Hobbytron* 2007). The cheapest device found was Tower Of Hanoi Kit for $29.75 USD as it's sold as a learning tool not as laboratory equipment (*Robotic Arm System* 2007). Commercial grade robotic arms are available for around $8000.00 USD (*180 Degree, 5 or 3 Finger Robot Arm* 2007).

### 3.1.3 Digital Switch

A digital switch provides the users with the ability to turn on and off a circuit. A digital switch can be easily connected directly to many different applications making this alternative very flexibly. Digital switches can be implemented by relays, Phototransistor Optocouplers, and transistors.

A relay is a small electronic device that uses magnetism to connect or disconnect two internal terminals in the relay. These two terminals connect to external circuitry allowing the relay to control whether the circuit is open or closed. Relays operate on different voltages, typically ranging from 300 to 600 volts AC with the electromagnet operating on voltages ranging from 24 to 120 volts AC. Relays also come is several different configurations including single pole single throw, double pole single throw, single pole double throw, double pole double throw, and quadruple pole double throw (*Relay - Wikipedia* 2007). Due to these features offered by relays they are a flexibility method of implementing a digital switch.

A transistor is a small current controlled semiconductive device that is used to amplify an electric current or as an electronic switch. The transistor works by applying a base/emitter voltage which increases the base/emitter current which allows for an exponential increase of the collector/emitter current (*Transistor - Wikipedia* 2007).

Phototransistor Optocouplers are a device which contains an infrared emitting diode and a NPN phototransistor. This device can operate in two modes; active and switch. In active mode the NPN transistor creates a response that is proportional to the light intensity received on the device until the phototransistor becomes saturated. In switch mode the phototransistor will either be on or off depending on the light received. Optocouplers are mainly used to isolate the input and output circuits from each other usually to protect the output circuit from damage caused by things such as surges and to help remove noise (*Optocoupler - Wikipedia* 2007).

### 3.1.4 Conclusion

| Method | Flexibility | Cost | Ability to complete task | Safety |
|---|---|---|---|---|
| Robotic Arm | ⋆⋆⋆⋆⋆ | ⋆ | ⋆⋆⋆⋆ | ⋆⋆⋆ |
| Digital Switch | ⋆⋆⋆⋆ | ⋆⋆⋆⋆⋆ | ⋆⋆⋆ | ⋆⋆⋆⋆ |
| Analog Switches | ⋆⋆⋆⋆ | ⋆⋆⋆⋆⋆ | ⋆⋆⋆⋆ | ⋆⋆⋆⋆ |

Table 3.1: Laboratory Interface Methods Summary

The robotic arm provides the user with the most flexibly means in which to manipulate the experiment and the fastest setup of new experiments. Unfortunately it is too expensive to implement for this research project and as such will no longer be considered a viable option.

Providing the user with access to both digital and analog switches will allow the users to best operate the laboratory equipment. Using such a combination will allows student to not only turn equipment on and off, but also adjust speeds, voltages, resistance, and other variable devices. Connecting these devices to a servomechanism will allow users to set levels on various laboratory equipment easily ensuring that experiments are completed quickly without users having to spend large amounts of time setting up experiment parameters, like a 12 volt input voltage.

## 3.2 Computer Interface Method

The Laboratory Equipment interface will need to be connected to a computer in order for it to work correctly. This can be done in several ways.

### 3.2.1 USB

USB stands for Universal Serial Bus. USB 1.1 and is capable of transferring up to 12Mpbs. A singular USB port can have up to 127 devices connected to it and is hot swappable. USB 2.0 is capable of transferring up to 480Mbps (*USB* 2007). USB 2.0 devices are backwards compatible with USB 1.1, meaning a USB 2.0 device can be connected to a USB 1.1 computer and a USB 1.1 device can be connected to a USB 2.0 computer. Currently all new Asus and Gigabyte motherboards are equipt with at lease 2 USB ports which are normally built directly onto the motherboard itself so that devices may be connected directly to the back panel (*Asus Motherboard* 2007) (*Gigabyte Motherboard* 2007).

### 3.2.2 FireWire/IEEE-1394

FireWire or IEEE-1394 can transfer at 3 different speeds, 100, 200, and 400Mbps. FireWire 800 or IEEE-1394b is capable of transferring up to 3.2Gbps (Axelson 2005). FireWire ports are sometime shipped with new computers. Most new Asus and Gigabyte motherboards are equipt with one FireWire port which is accessibly from the rear panel. They also come with two connectors in which external cabling and connectors are used to allow FireWire connectivity (*Asus Motherboard* 2007) (*Gigabyte Motherboard* 2007). In some instances a PCI, PCMCIA or USB FireWire card needs to be purchased to interface with the FireWire device. FireWire devices are generally more expensive then USB and are best suited to applications where high communication speed is essential or the device needs to broadcast to multiple receivers (Axelson 2005).

### 3.2.3 Serial Port

Serial ports are capable of data transfer speeds of up to 20kbps (Axelson 2005). Serial ports are equipt on most new Gigabyte boards however the new Asus motherboards no longer have a serial port build onto the motherboard (*Asus Motherboard* 2007) (*Gigabyte Motherboard* 2007). In instances such as this a PCI, PCMCIA or USB to Serial adapter card needs to be purchased to interface with the serial device.

### 3.2.4 Parallel Port

Parallel ports are capable of transferring up to 8Mpbs (Axelson 2005). Parallel ports are equipt on most new Gigabyte boards however the new Asus motherboards no longer have a parallel port built onto the motherboard (*Asus Motherboard* 2007) (*Gigabyte Motherboard* 2007). In instances such as this a PCI, PCMCIA or USB to Parallel adapter card needs to be purchased to interface with the parallel device.

### 3.2.5 Conclusion

| Method | Speed | Cost | Availability |
|---|---|---|---|
| USB | ⋆⋆⋆⋆ | ⋆⋆⋆⋆⋆ | ⋆⋆⋆⋆⋆ |
| FireWire/IEEE-1394 | ⋆⋆⋆⋆⋆ | ⋆⋆⋆ | ⋆⋆⋆⋆ |
| Serial Port | ⋆ | ⋆⋆ | ⋆⋆ |
| Parallel Port | ⋆⋆ | ⋆⋆ | ⋆⋆ |

Table 3.2: Computer Interface Methods Summary

All of these computer interface methods can easily be acquired via the purchase of an add-on card should the computer not have the required functionality. Although doing so increases overall cost of the system. Interface methods such as USB and FireWire appear to be replacing serial and parallel with the release of USB printers and scanners, and the removal of these ports from some manufacturers motherboards.

The communication speed required by the interface is very small and as such the increased speed benefit, offered by FireWire, is not offset by the increased cost. A USB device sufficiently provides an easy interface for the device to connect to the computer and adequate speed for communication.

## 3.3 Switches

In order for the Remote Laboratory to be able to operate all of the possible experiments from Tables 1.2 to 1.5 the Remote Laboratory needs to provide 4 digital and 4 analog switches.

### 3.3.1 USB Relay Alternatives

| Product Name | Connectors | Programming Languages |
|---|---|---|
| ADU200 | USB, RS232 | VB, C++, Borland, Java |
| ADU208 | USB, RS232 | VB, C, C++, C#, Borland, Java |
| ADU218 | USB, RS232 | VB, C, C++, C#, Borland, Java |
| Extended USB Interface Board | USB | VB, C++, Delphi |
| LabJack U3 | USB | VB, C, C++, Borland, Java, Matlab |
| USBREL8 | USB | VB, C++ C#, Delphi, Java |
| USB ProXR | USB, RS232 | VB |
| USB Experimenter's Interface Kit | USB | ? |

| Product Name | Contact Details | Website |
|---|---|---|
| ADU200 | tfortin@vianet.on.ca | http://www.ontrak.net/ |
| ADU208 | tfortin@vianet.on.ca | http://www.ontrak.net/ |
| ADU218 | tfortin@vianet.on.ca | http://www.ontrak.net/ |
| Extended USB Interface Board | info@vellemanusa.com | http://www.vellemanusa.com |
| LabJack U3 | info@labjack.com | http://www.labjack.com |
| USBREL8 | sales@quancom.de | http://www.quancom.de/ |
| USB ProXR | tech0001@control anything.com | http://www.controlanything.com/ |
| USB Experimenter's Interface Kit | | http://www.jaycar.com.au |

Table 3.3: Relay Manufacturers

| Product Name | Manufacturer | Output Ports | Input Ports | Cost |
|---|---|---|---|---|
| ADU200 | Ontrak Control Systems | 4 x 120VAC or 30VDC at 5A | 4 x TTL Digital Inputs | $139US + $60US P& H |
| ADU208 | Ontrak Control Systems | 8 x 120VAC or 30VDC at 5A | 8 x TTL Digital Inputs | $189US + $60US P& H |
| ADU218 | Ontrak Control Systems | 8 x 120VAC or 30VDC at 5A Solid State | 8 x TTL Digital Inputs | $225US + $60US + P& H |
| Extended USB Interface Board | Velleman Inc | 8 x Open Collector and 8 x 8 bit Analog | 8 x Open Collector and 8 x 8 bit Analog | $124.95US + P& H |
| LabJack U3 | LabJack | 20 x high low and 1 8 bit Analog | 16 x Digital or analog | $99USD + P& H |
| USBREL8 | Quancom | 8 x 30V at 1A (Plug in 4 devices) | None | 117.81 Euro + P& H |
| USB ProXR | National Control Devices | 8 x ?V at 5A (Plug up to 256 relays) | 10 bit A/D | $186US + P& H |
| USB Experimenter's Interface Kit | Jaycar Electronics | 8 x Open collector digital | 5 x Digital 2x Analog | $62.95AU + P& H |

Table 3.4: Relay Specifications

### 3.3.2 Conclusion

| Method | Output Ports | Cost | Programming languages |
|---|---|---|---|
| ADU200 | ⋆ | ⋆⋆⋆ | ⋆ |
| ADU208 | ⋆⋆⋆ | ⋆⋆ | ⋆⋆⋆⋆⋆ |
| ADU218 | ⋆⋆⋆ | ⋆ | ⋆⋆⋆⋆ |
| Extended USB Interface Board | ⋆⋆⋆⋆⋆ | ⋆ | ⋆⋆⋆ |
| LabJack U3 | ⋆⋆⋆⋆ | ⋆ | ⋆⋆⋆⋆⋆ |
| USBREL8 | ⋆⋆ | ⋆⋆ | ⋆⋆⋆⋆ |
| USB ProXR | ⋆⋆ | ⋆⋆⋆ | ⋆ |
| USB Experimenter's Interface Kit | ⋆⋆ | ⋆⋆⋆⋆⋆ | ? |

Table 3.5: USB Relay Alternative Summary

The Extended USB Interface Board by Velleman Inc provides ample digital and analogue switches for the current practical courses, while the other providers do not offer enough switches. The Extended USB Interface Board also provides enough switches to accommodate the possible development of new experiments which may be more sophisticated.

## 3.4 User Feedback

The user will need to be provided with some level of feedback to show them how the experiment reacted to the instructions issued.

### 3.4.1 Video

Video feedback provides the most detailed method of portraying to the user what occurs in the laboratory. Windows Media Encoder is freely available for download from Microsoft's homepage and offers an easy implementation of both audio and video feedback. Through trials of running this program the bandwidth required to transmit a video and audio signal ranges from 2000 kbps to 12 kbps depending on video size, quality, and frames per second and the audio quality, and number of audio channels. Video and audio streaming does however require some encoding and decoded and from experience with these programs it was reveled that there is a delay in the video feed, usually of about 10 seconds. A delay of this magnitude can make performing experiments very difficult and slow especially when a particular event occurs very quickly.

### 3.4.2 Still Images

User feedback can be achieved by using a series of still images that are captured from a camera. The frequency in which these still images are taken determines whether the images look like a video feed or just a picture. A program such as Fwink (*Fwink* 2007) can be used to read the images from the camera and upload them to an FTP server or a local directory where they can be displayed to the user. Alternatively a similar sub-program could be coded to read the feed from the video camera and save them as a file to be displayed to the user. Allowing the user to control the frequency of these images would allow users on slower connections, such as dialup, to be able to perform these experiments.

### 3.4.3 Audio

Audio feedback can be useful in conveying to the user what occurred during the experiment. Deakin University in Melbourne currently has a manufacturing laboratory which is controlled remotely. They provide audio and visual feedback to their students so that they can hear the noises that the machine makes when it cuts the material (Wong, Ferguson, Florance, Bantwal & Jones 2007). Unfortunately audio streaming is also subject to the delays incurred in video streaming.

### 3.4.4 Software physical event

The experiment could be designed so that when an special event occurs in the laboratory a signal is received by the host computer. When this signal is received by the host computer it triggers an event to be displayed to the user, such as a saved image, playback of a sound, or any other event that the user could identify.

### 3.4.5 Software event

In this approach the remote laboratory has several pre-programmed events which, when activated by the user entering certain information, will display an image to the user.

### 3.4.6 Conclusion

| Method | Bandwidth | Realtime | Flexibility | Cost | Portrayal |
|---|---|---|---|---|---|
| Video | ⋆⋆ | ⋆ | ⋆⋆⋆ | ⋆ | ⋆⋆⋆ |
| Still Images | ⋆⋆⋆ | ⋆⋆⋆ | ⋆⋆⋆ | ⋆ | ⋆⋆⋆ |
| Audio | ⋆⋆⋆ | ⋆ | ⋆ | ⋆⋆ | ⋆⋆ |
| Software physical event | ⋆⋆⋆ | ⋆⋆⋆ | ⋆⋆⋆ | ⋆⋆⋆ | ⋆⋆ |
| Software event | ⋆⋆⋆⋆⋆ | ⋆⋆⋆⋆⋆ | ⋆⋆⋆ | ⋆⋆⋆ | ⋆ |

Table 3.6: Feedback Methods Summary

Video and audio feedback are currently the most commonly used feedback types used in remote laboratories. These methods are also the most effective at conveying to the user what occurred in the experiment. Unfortunately the delays incurred with video streaming make it an unusable option in some experiments. Software event and Software physical event methods do not portray to the user what actually occurred in the experiments because their responses are preprogrammed to a given input. The still images approach provides the user with an accurate portrayal of what occurred in the laboratory and by allowing the user to adjust the frame rate will allow dialup users to use the system similar to the speeds that broadband users do.

# Chapter 4

# Software Requirement Alternatives

## 4.1 Information Storage

The Information storage method is used to store information about user, experiments, and bookings. This operation can be performed by Domains, Access Databases, My SQL Databases, Text files, or hard coding the information in. In some situations it is also possible to utilise an existing database.

### 4.1.1 Domain

The use of a new or existing domain controller could be used to store data for use on the remote laboratory software. Access to an experiment could be restricted to two domain groups, one for students and one for administrators. When a student is authorised to access the system they are added to the domain group for students. Once the student has completed the experiment they are removed from the domain group so they are no longer authorised to access the experiment.

### 4.1.2 Existing database

The Remote Laboratory may be able to utilise an existing database that is currently being used at the university. This would reduce the amount of redundant information because the existing database would contain information such as name, user name, and a password. The remote laboratory system could then connect to the existing database to gain this information for authentication purposes. In addition to reducing the amount of redundant data this method also provides the advantage of students using their current university passwords. Using this method does present problems with portability as the system would need to be redesigned specifically for each existing database used.

### 4.1.3 Microsoft Access database

A Microsoft Access database could be used to store all the information about the users and experiments that are to be run in the remote laboratory. This method allows the remote laboratory to be completely independent from any existing system which would allow this system to still operate even when a complete redevelopment of existing databases and websites is to be performed. Access is not a true client, server database program and as such Microsoft Access databases are best implemented in situation where the database size doesn't exceed 2 Giga-bytes. Although many comparison charts indicate that Microsoft Access can operate with 255 concurrent users an article published on Microsofts stated:

> "Jet can support up to 255 concurrent users, but performance of the file-based architecture can prevent its use for many concurrent users. In general, it is best to use Jet for 10 or fewer concurrent users."

(*MSDE for Microsft Visual Studio 6.0: An Alternative to Jet for Building* 2007)

Microsoft Access provides a relatively inexpensive method of implementing a database because is is part of the Microsoft Office packages currently available (*Benefits and Constraints of using Microsoft Access Database & Office* 2007).

### 4.1.4 MSDE database

An MSDE database provides a database alternative similar to the specification of a Microsoft Access Database. The architecture of an MSDE database differs from Access in that a separate program is used to interface with the database. This allows for faster data access and more concurrent connections. The MSDE database system was designed for small scale use and as such allows for only five concurrent users but has the advantage of using a smaller amount of system resources (*What are the capacities of Access, SQL Server, and MSDE?* 2007).

### 4.1.5 MySQL

MySQL is a database management alternative that is available on many different platforms for substantially less then Oracle and SQL Server. Implementing a MySQL server is free for private and educational use. Websites such as Wikipedia have adopted MySQL as their database engines (*SQL Server vs MySQL* 2007). Features that are standard to SQL Server such as triggers, stored procedures, and foreign keys have not been fully implemented until the most recent release of MySQL 5.X. MySQL databases generally perform better then SQL Server database because of the default table format of it's MyISAM database, which is both compact and uses less resources then SQL Server (*MySQL 5.0 vs. Microsoft SQL Server 2005* 2007). MySQL has also been found to read data very fast, however updating of records is slower then SQL Server (*Choosing The Right Programming Language* 2007).

### 4.1.6 SQL Server database

SQL Server was designed to be a robust database management system using similar architecture to MSDE and as such provides support for 32,767 concurrent user connections (*Benefits and Constraints of using Microsoft Access Database & Office* 2007). Microsoft SQL Server is available in several different versions to suit the specific clients database needs. The Express edition of SQL Server is available free at Microsoft's homepage. SQL Servers greatest benefits is that it provides a broad range of native data analysis and reporting tools. Theses tools do however degrade the overall performance of the system (*What are the capacities of Access, SQL Server, and MSDE?* 2007).

SQL Server comes with reporting services which are widely used. Other database alternatives, such as MySQL, don't come with these reporting tools which can be purchased from a third party provider at an additional cost (*SQL Server vs MySQL* 2007). SQL Server has been certified as C-2 compliant, which means that SQL Server is suitable for use in government applications because it contains the appropriate security requirements (*MySQL 5.0 vs. Microsoft SQL Server 2005* 2007)

### 4.1.7 Oracle database

An Oracle Database provides similar features and performance to a SQL Server database however it is substantially more expensive to implement, costing at least $10,000 more for the Standard version then an equivalent SQL Server. Oracle is considered by most to be more difficult to setup and configure then equivalent SQL Servers. An Oracle database does, provide that advantage of being platform independent and fine tuning can be implemented using start up parameters (*SQL Server vs Oracle* 2007).

### 4.1.8 Textfile

Text files are a common method for storing information. User authentication details can be stored and retrieved quickly and simply, and require no special software to be installed or running. Textfile storage is able to be viewed and changed by any ASCII file viewer which leaves a security hole in the system. Any user who can view this textfile is able to see all the user names and passwords that allow access to the system.

### 4.1.9 Written into the code of the program

Writing the database into the code of the program, or hard coding, provides little to no flexibility. In doing so most programming languages require a recompiling of the source files in order to reflect changes. This is not only time consuming but requires constant interaction from the writer of the software to implement changes in the system. This method would only be used as a last resort.

### 4.1.10 Conclusion

| Method | Cost | Portability | Redundancy | Changeability | Security |
|---|---|---|---|---|---|
| Domain | ⋆⋆⋆ | ⋆⋆⋆⋆⋆ | ⋆⋆⋆⋆ | ⋆⋆⋆⋆ | ⋆⋆⋆⋆⋆ |
| Existing DB | ⋆⋆⋆⋆ | ⋆ | ⋆⋆⋆⋆ | ⋆ | ⋆⋆⋆ |
| Access DB | ⋆⋆⋆⋆ | ⋆ | ⋆⋆⋆ | ⋆⋆⋆⋆⋆ | ⋆⋆⋆⋆ |
| MSDE | ⋆⋆⋆⋆ | ⋆ | ⋆⋆⋆ | ⋆⋆⋆⋆⋆ | ⋆⋆⋆⋆ |
| MySQL | ⋆⋆⋆⋆⋆ | ⋆⋆⋆⋆⋆ | ⋆⋆⋆ | ⋆⋆⋆⋆⋆ | ⋆⋆⋆⋆ |
| SQL Server | ⋆⋆⋆ | ⋆ | ⋆⋆⋆ | ⋆⋆⋆⋆⋆ | ⋆⋆⋆⋆ |
| Oracle | ⋆ | ⋆⋆⋆⋆⋆ | ⋆⋆⋆ | ⋆⋆⋆⋆⋆ | ⋆⋆⋆⋆ |
| Text File | ⋆⋆⋆⋆⋆ | ⋆⋆⋆⋆⋆ | ⋆⋆⋆ | ⋆⋆⋆ | ⋆⋆⋆ |
| Hard Code | ⋆⋆⋆⋆⋆ | ⋆⋆⋆⋆⋆ | ⋆⋆⋆ | ⋆ | ⋆⋆⋆⋆⋆ |

Table 4.1: User Authentication Methods Summary

There are many database products on the market and within this project several of the popular methods have been analysed.

Due to current security restrictions on the university network the domain and existing database methods shall not be used. Using textfiles to store information (particularly relational databases) can be very difficult to code, maintain, and troubleshoot. A database alternative provides an easier and cleaner method to store information.

The Oracle database does provide a stable alternative to the Microsoft products however it comes at a substantially higher cost. The increased cost outweighs the advantages, such as portability, as it would be cheaper to purchase a Windows server and an equivalent version of SQL Server then it would be to purchase the Oracle database solution.

The Microsoft Access and MSDE alternatives provide very similar specification with architecture the main difference. In the event that the resources required to run the remote laboratory software exceed the capabilities of both Access and MSDE the database system will need to be up-sized to a SQL Server installation. In this case it is easier to up-size an MSDE database then an Access one because of the similar architecture used in both MSDE and SQL Server.

A MySQL database provides a cost-effective alternative to Oracle and SQL Server. It is a command prompt based program which can be more difficult to setup then the Microsoft equivalent. Due to the portability and cost effectiveness of a MySQL database alternative this shall be implemented for use in the remote laboratory.

## 4.2 Booking System

The booking system is used to allow students to book times that they wish to use the remote laboratory. The user will be able to log into the system, see the times that the remote laboratory will be available, and book an unallocated time. The booking system can be added into the information storage system to stop people who aren't booked in from controlling the hardware.

### 4.2.1 Single User System

A single user system would be designed to only let one user access the system at a time. The system has no need for communication between users and all experiments would need to be programmed for individual work.

### 4.2.2 User, Viewer System

In the user, viewer system there is only one person who can manipulate the operations of an experiment. The system would also allow for one other person to view the system, possibly an examiner watching the experiment to observe and assess that the experiment is being performed correctly, safely, and so that the outcomes are achieved. Communication between the user and the viewer would be of benefit especially in explaining why something was done or occurred during the course of the experiment.

### 4.2.3 Multi-User System

A multi-user system would allow a team of students or examiners to interact with the experiment creating an experience more like the atmosphere at residential school. The experiment could be controlled via a token system, whoever holds the token has the ability to manipulate the experiment as they wish. The token may be passed to another student as required. Users would need to communicate with each other and this could be achieved via an audio system however this would require that all the users have access to fast connections, speakers, and a microphone. A more appropriate solution would be to have a text based chat program so that all user can communicate, even those with slow Internet connections. As the participation of a group experiment is difficult to judge the experiment could be broken into tasks and logs could be used to record the actions of each user to ensure that all members are participating. However, it is difficult to ensure that all users are participating fully when they are operating as a group as other members can instruct an individual on what to do to complete their section. Secondly, the other members of a group who are not controlling the experiment may become inattentive or disruptive. Under these circumstances it may be better to implement one of the other alternatives.

### 4.2.4 Conclusion

| Method | Flexibility | Complexity | Determination of User Participation | Student Interaction |
|---|---|---|---|---|
| Single User | ⋆ | ⋆ | ⋆⋆⋆⋆⋆ | ⋆ |
| User, Viewer | ⋆⋆ | ⋆⋆⋆⋆⋆ | ⋆ | ⋆⋆ |
| Multi User | ⋆⋆⋆⋆⋆ | ⋆⋆⋆⋆⋆ | ⋆ | ⋆⋆⋆⋆⋆ |

Table 4.2: Booking Systems Summary

The single user booking system provides an uncomplicated method in which individual students can perform a remote laboratory experiment and be clearly seen to have participated in the experiment. However, it does remove one of the key elements of attending the university campus which is student interaction. The multi-user and the user, viewer methods are equally as complicated to implement, however the multi-user system provides substantially more student interaction and cooperation while the user viewer section does only allows an additional user to view the session to see how it works.

A booking system which employs the concepts found in all three methods will be used in this project. Experiments will be setup stating how many users may work together to complete the experiment. This allows for several people to work on an experiment at once but at the same time an experiment may be setup to only allow one user at a time to participate. One of the operators will have a control token which allows them to manipulate the experiment as they see fit. The other users may invoke an election in which all users will vote on who will have the control token to stop one user from holding control. The system will allow the administrator to assign users to a group were they will be able to perform and observe the experiment. Administrators will be able to interact with any operational remote laboratory and may take the control token. Users will be able to communicate with each other using a text based system where they will type a message and the other users will be able to see and respond if desired.

## 4.3 Programming Languages

### 4.3.1 ASP

ASP is a programming language used on many websites throughout the world and as such there is no language on the web better supported. ASP is an interpreted scripting language which means that the code used in ASP pages is interpreted line by line, each time that they are run. ASP websites have the advantage of being low cost because the software require to host these pages comes with every Microsoft Windows Server and the ASP pages can be written in any text programs, such as Microsoft Wordpad and Notepad (*Choosing The Right Programming Language* 2007).

### 4.3.2 ASP.Net

ASP.Net is a newer version of ASP. ASP.Net pages are not interpreted scripting pages like ASP, but are part of the DotNet Framework. ASP.Net pages contain scripts that are automatically compiled by the hosting server (similar to how C++ programs are compiled). The pages are then stored in memory for the next visitor which makes the compilation of ASP.Net pages a lot faster then ASP. ASP.Net is an object oriented programming language which offers a unique way of triggering code on the server from web pages in the client's web browsers. ASP.Net offers Membership and Role provider access with standardised login controls that can be easily inserted onto web forms as well as intelligent caching and garbage collection (*Choosing The Right Programming Language* 2007).

### 4.3.3 C

C and C++ programming languages is a multi-platform programming language that supports procedural programming, data abstraction, object-oriented programming, and generic programming. C and C++ programming languages are commonly found running on Windows, Linux, and UNIX systems and appear to be an ideal language to develop the host program because of it's strengths and simplicity (*C - Wikipedia* 2007). C and C++ are coded in any text program then complied using the appropriate compiler for the system that it will run on.

### 4.3.4 ColdFusion

ColdFusion is a programming language for websites that consists of a series of predefined tags that do server side instructions, such as connecting to a database. Unfortunately there is a predefined number of tags and some tasks require a large amount of processing, which can be done easier in a different language. The software required to host a ColdFusion website is not as common as others researced, however there are still many hosting companies that support ColdFusion sites (*Choosing The Right Programming Language* 2007).

### 4.3.5 Java

Currently there are several remote laboratories that are under redevelopment that were written in Java. It was found that Java was having problems over the years and error messages were being received by the client computers making the system inoperable. Java does have the advantage of being both a website language and a desktop engine which allows for one programming language to be used throughout the project.

### 4.3.6 PHP

PHP is the programming language of choice for many Linux users. It is free to host on both Windows and Linux operating systems. PHP is an interpreted scripting language making it similar to the operation of ASP and is mostly used with MySQL database systems (*Choosing The Right Programming Language* 2007).

### 4.3.7 Visual Basic

Microsoft's Visual Basic is a programming language that has been developed by the Microsoft Corporation to allow graphical user interface (GUI) programs to be developed quickly and easily but to have the power and sophistication to perform complex operations. Visual Basic does have several disadvantages. Being developed for the Windows operating system it is not compatible with Linux or Unix, although compilers for these systems are being developed (*Mono brings Visual Basic Programs to Linux* 2007). It is not a proper object oriented language nor does it support threading of processes (*Visual Basic - Wikipedia* 2007). Visual Basic, as the name suggests, is a easy language to learn and start with, making it suitable for the host program.

### 4.3.8 Conclusion

| Method | Ability perform the task | Requirements | Portability | Cost |
|---|---|---|---|---|
| ASP | ⋆⋆⋆⋆⋆ | ⋆⋆⋆⋆⋆ | ⋆⋆⋆ | ⋆⋆⋆ |
| ASP.Net | ⋆⋆⋆⋆⋆ | ⋆⋆⋆ | ⋆⋆⋆ | ⋆⋆⋆ |
| C | ⋆⋆⋆⋆⋆ | ⋆⋆⋆⋆⋆ | ⋆⋆⋆⋆⋆ | ⋆⋆⋆⋆⋆ |
| ColdFusion | ⋆⋆⋆ | ⋆⋆⋆ | ⋆⋆⋆⋆⋆ | ⋆⋆⋆ |
| Java | ⋆⋆⋆⋆⋆ | ⋆⋆⋆⋆⋆ | ⋆⋆⋆⋆⋆ | ⋆⋆⋆⋆⋆ |
| PHP | ⋆⋆⋆⋆⋆ | ⋆⋆⋆⋆⋆ | ⋆⋆⋆⋆⋆ | ⋆⋆⋆⋆⋆ |
| Visual Basics | ⋆⋆⋆ | ⋆⋆ | ⋆ | ⋆⋆ |

Table 4.3: Programming Languages Summary

Java provides a solution that would only require the knowledge of one programming language, however with several remote laboratories currently using Java and not working this could be a risky move and as such will not be used for this project.

ColdFusion provides an easy language to get started in website development but the hosting of this can be expensive. ASP and PHP provide similar alternatives to developing a website but PHP is free. Although PHP provides a free solution for the hosting of websites ASP.Net utilises the latest Dot Net Framework technology. This programming language is personally more familiar and as such will be used for the development of the Website interface with this research project.

C and C++ provide a portable solution for the host program so this will be used as Visual Basics currently can only be run on a Windows operating systems.

# Chapter 5

# Required System Specification

## 5.1 Assumptions

Currently there are 26 experiments from 4 different subjects with experiments that can be done in a remote laboratory and it is assumed that this will grow to 50 experiments from 10 different subjects with, at most, 5 students interacting with each experiment. It is also assumed that 1 lecturer from each course will be able to log into the system to observe the experiments. Therefore the system should be able to cope with 260 users concurrently being logged in.

Several of these experiments perform the same tasks as another experiment but they are for a different subject so the focus of the experiment might be slightly different and as such these experiments will be considered as a different experiment.

The following assumptions have been made:

- Requirement for laboratory will be 4 digital switches and 4 analog switches.
- Growth rate will be 10% per year for 10 years before the system will be redeveloped.
- 260 concurrent users could be logged in.
- The university is responsible for identifying students and providing them with unique student ID's

## 5.2 Software Operation

The required software operations are to:

- Implement a booking system for students to book the laboratory for use.
- Allow the authenticated users to change states of an experiment.
- Display to authenticated users what has occurred in the laboratory.
- Monitor the users access to insure:
  - Safety of the user
  - Safety of the laboratory equipment
  - Only authenticated users can access the laboratory
  - Duration of time the user has logged onto the system
  - A user isn't idly logged on
- Provide easy reconfiguring for different experiments.
- Ability to save the feedback shown to the user.
- Provide 4 digital and 4 analog switches for users to change.
- Provide access for 260 concurrent users.
- Provide adequate proof to course assessor that the experiment was completed correctly.

Proof of completion shall be shown to the examiners by a series of screen shots that are taken by the user when they request it to be taken.

Web-page based design shall be used to allow the user to interface with the remote laboratory system.

## 5.3 Interface Program Operation

When the website is opened it should prompt the user for a user name and password, and display any messages to the users. After logged in the user will then see either the Administrator or Student interface.
The Administrator Interface will contain the ability to:

- Add/edit users.
- Enable/disable a user to use the system.
- Add/edit bookings for any user.
- Create/change an experiment
- Enable/disable an experiment from use.
- Change the time a student can spend on an experiment.
- Join any session to view and chat with the student/s.

The Student Interface will contain:

- The ability to change password.
- A description for each experiment.
- The available times for all of the laboratories that they may enter.
- The amount of time that they have left working on a remote laboratory.
- Any remote laboratories that they may enter. The user will only be able to enter a remote laboratory if:
  - The experiment is functional.
  - The experiment has started.
  - The experiment has not finished.
  - The student has time allocated to that experiment.
  - They have previously booked for the current time.
  - The laboratory isnt booked and not in use by another user.

After the user enters an experiment it displays:

- Time remaining.
- Instructions on the experiment.
- Any available changes that can be made.
- Pause video feed button.
- Save image button.
- Chat interface

After the user enters a change it is appended to the database which is periodically read by the host computer which implements the changes in the remote laboratory. When the user believes that the experiment is completed a collection of saved images is emailed to the course assessor as proof that the experiment was completed correctly.

## 5.4 Host Program Operation

The host computer will run a program as a service or daemon that is responsible for the interaction with the laboratory. It will periodically contact the database and download a copy of the actions to be implemented in the remote laboratory. Any changes to the states of the switches are then implemented. The host computer software will also upload a still image from the video camera/s for use in user feedback. The host computer is also responsible for monitoring the frequency of changes to avoid damage to the equipment. Once the changes have been implemented in the remote laboratory they are deleted from the database.

## 5.5 Database Structure

The database will contain two main tables; the User and Experiment tables. The records entered into these tables should not be deleted, but if necessary they can be disabled to stop a user logging in, or stop an experiment from running. This ensures that the database keeps its integrity and can be done with a boolean field called 'Enabled' which is set to false.

In a relational database tables are linked together. This reduces redundant code but does create problems during the update and deletion on records. To overcome this problem extra conditions are enforced on the database to maintain integrity. The two main options are; cascade update and cascade delete. Cascade update is valuable in that when a record is updated, any records that link to that record are updated as well. Applying cascade updates increases update time but ensures that no references are left hanging. Cascade delete is used to remove hanging references when a record is deleted. It does this by deleting all other records that reference the deleted record. When cascade delete is not used, and the delete command is issued, an error message will occur if the record is being referenced by another record.

ID fields will be using in some tables as a unique identifier for the records. This is done because it is faster to perform updates when tables are linked by a static field. In situations where tables are joined together using a name field for example, when an update to that name is issued, all records referencing that name also need to be updated, whereas if the tables were linked by an ID field and the name was updated, then only one record is updated. The ID fields will mainly be autonumbers.

The remote laboratory software for this projecg will use cascade update but not cascade delete. This will ensure that records are updated and that should a referenced record be requested to be deleted an error message will be generated. A complete copy of the code required to create the database is found in Appendix B.

### 5.5.1 Membership and Role Provider Tables

ASP.Net 2.0 has implemented a scheme to authenticate users called the Membership and Role Providers. This schema provides web developers with an easy way to secure their websites because it performs all the tasks required to secure a website behind the scenes. The Membership and Role Provider was originally implemented to work with MS SQL Server and Microsoft Access but the developers of MySQL have created a connector called 'Connector/Net 5.1' which, as of 5th of September 2007 is still a beta release, allows MySQL database to be used as a Membership and Role Provider (*MySQL AB* 2007).

The following MySQL membership table, roles table, and users in roles table are created by using a Membership and Role Provider in the MySQL Database.

#### MySQL Membership Table

The MySQL Membership table contains the information about the user, such as user name, password, and statistics. This table was designed so that when new users enter a website they can register and then log in without any interaction with the site administrator. It also provides functionality for users to; reset their password, cancel their account, and see which other users are currently online. The 'comment' field will be used to store the users first and last names. This is done because the MySQL Membership table does not have first and last name fields. Table 5.3 shows all the information stored in the mysql_membership table.

### MySQL Roles Table

The MySQL Roles Table contains two fields. The first field, Rolename, specifies the name of a role. The role name is often granted or denied permissions on a directory, or file of the website. This is useful in stopping unauthorised users from performing functions but allowing authorised users to perform these actions. This table also stores the application name. This allows one singular membership database to be used for several different applications. Table 5.1 shows all the information stored in the mysql_roles table.

### MySQL Users In Roles Table

The Users in Roles table is used to create a many-to-many join on the Membership and Roles tables. This allows for many users to be in many different roles and many roles to belong to many different users. This table contains three fields; Username and Role name, to create the many-to-many join, and the application name, so that one membership database can be used for many different applications. Table 5.2 shows all the information stored in the mysql_usersinroles table.

| Field | Type | Null | Key | Default |
|---|---|---|---|---|
| Rolename | varchar(255) | NO | MUL | |
| ApplicationName | varchar(255) | NO | | |

Table 5.1: mysql_roles table

| Field | Type | Null | Key | Default |
|---|---|---|---|---|
| Username | varchar(255) | NO | MUL | |
| Rolename | varchar(255) | NO | | |
| ApplicationName | varchar(255) | NO | | |

Table 5.2: mysql_usersinroles table

| Field | Type | Null | Key | Default |
|---|---|---|---|---|
| PKID | varchar(36) | NO | PRI | |
| Username | varchar(255) | NO | | |
| ApplicationName | varchar(255) | NO | | |
| Email | varchar(128) | NO | | |
| Comment | varchar(255) | YES | | NULL |
| Password | varchar(128) | NO | | |
| PasswordQuestion | varchar(255) | YES | | NULL |
| PasswordAnswer | varchar(255) | YES | | NULL |
| IsApproved | tinyint(1) | YES | | NULL |
| LastActivityDate | datetime | YES | | NULL |
| LastLoginDate | datetime | YES | | NULL |
| LastPasswordChangedDate | datetime | YES | | NULL |
| CreationDate | datetime | YES | | NULL |
| IsOnline | tinyint(1) | YES | | NULL |
| IsLockedOut | tinyint(1) | YES | | NULL |
| LastLockedOutDate | datetime | YES | | NULL |
| FailedPasswordAttemptCount | int(10) unsigned | YES | | NULL |
| FailedPasswordAttemptWindowStart | datetime | YES | | NULL |
| FailedPasswordAnswerAttemptCount | int(10) unsigned | YES | | NULL |
| FailedPasswordAnswerAttemptWindow Start | datetime | YES | | NULL |

Table 5.3: mysql_membership table

### 5.5.2 Experiment Table

The experiments table is used to store information about the experiments so contains a name, experiment description, and experiment instructions. In order for students to book a time to complete the experiment it is important to know the maximum running time needed to completed the experiment and the number of operators that may use the system at once. To reference a record uniquely an ID field is used. Table 5.4 shows all the information stored in the experiments table.

| Field | Type | Required | Key | Unique |
|---|---|---|---|---|
| ID | autonumber | Yes | Yes | Yes |
| Name | varchar(30) | Yes | | Yes |
| Description | varchar(256) | No | | |
| Instructions | text | No | | |
| TimeReq | time | Yes | | |
| BookingSlot | time | Yes | | |
| NoOfOps | tinyint (unsigned) | Yes | | |
| IsGroupBookable | boolean | Yes | | |
| Enabled | boolean | Yes | | |

Table 5.4: experiments table

### 5.5.3 User Grouping

There are currently several experiments that may be completed as a group. As an additional part of the database the system administrators will be able to assign users to groups. This is an optional setting which will not influence the operation of the remote laboratory if not used. The creation of user groups requires a many-to-many relationship between the Users and Groups tables.

### Group Table

The group table stores the name of a group and an ID field. Table 5.5 shows all the information stored in the groups table.

| Field | Type | Required | Primary Key | Unique |
|---|---|---|---|---|
| ID | autonumber | Yes | Yes | Yes |
| Group Name | varchar(20) | Yes | | Yes |

Table 5.5: groups table

### Group Users Table

The group users table stores the username of a member and the group ID of the group they are in. Both the username and group ID are primary keys which enforces that any one user can only be in a particular group once, but that one user can be in many groups, and that groups contain many users. The username field is linked to the username field in the MySQL Membership table and the group ID field is linked to the ID field in the Groups tables so that an update in any of these fields will update the other tables as required. Table 5.6 shows all the information stored in the groupusers table.

| Field | Type | Required | Primary Key | Unique |
|---|---|---|---|---|
| Username | varchar(255) | Yes | Yes | Yes |
| Group ID | int (unsigned) | Yes | Yes | Yes |

Table 5.6: groupusers table

### 5.5.4 Subject Grouping

Subject grouping will stop users who are not in a particular subject from performing an experiment for a subject that they are not studying. Users are added to the usersinsubects table so they can perform an experiment and once they have completed the subject they are removed so that they cannot perform the experiment again.

The creation of subject grouping requires 2 tables with a many-to-many relationship.

#### Subjects Table

The subjects table is used to store a list of all the subjects that have experiments that will be performed in the remote laboratory. The ID field will be used to store the subject code.

| Field | Type | Required | Primary Key | Unique |
|---|---|---|---|---|
| ID | varchar(20) | Yes | Yes | Yes |
| Name | varchar(20) | Yes | | Yes |

Table 5.7: subjects table

#### Users In Subjects table

The users in subjects table will store a list of the usernames of students and the corresponding subjects that they may perform experiments in.

| Field | Type | Required | Primary Key | Unique |
|---|---|---|---|---|
| UserName | varchar(255) | Yes | Yes | Yes |
| Subject ID | int (unsigned) | Yes | Yes | Yes |

Table 5.8: usersinsubjects table

### 5.5.5 Experiment Period Table

The experiment period table is used to create times which an experiment can be conducted. This table stores the experiment ID for which the time constrains apply to, the start date and time, and stop date and time. The experiment ID field is linked to the Experiments table ID field, so that an update on the Experiments table ID field will update the Experiment Period table. Table 5.9 shows all the information stored in the experimentperiod table.

| Field | Type | Required | Primary Key | Unique |
|---|---|---|---|---|
| Experiment ID | int (unsigned) | Yes | Yes | Yes |
| Start | Date/Time | Yes | Yes | Yes |
| Finish | Date/Time | Yes | Yes | Yes |

Table 5.9: experimentperiod table

### 5.5.6 Instructions Table

Changes to the experiment will be added to this table where they will be read by the host program and implemented on the equipment in the laboratory.

| Field | Type | Required | Primary Key | Unique |
|---|---|---|---|---|
| Switch ID | int (unsigned) | Yes | Yes | Yes |
| Experiment ID | int (unsigned) | Yes | Yes | Yes |
| Switch Value | tiny int | Yes | | |
| Time | Date/Time | Yes | Yes | Yes |
| UserName | varchar(255) | Yes | | |

Table 5.10: instructions table

### 5.5.7 Switches Table

The switches tabl is used to add switches to experiments. This table contains the Experiment ID and switch description. These fields are unique so that one experiment cannot have two switches with the same description. This table also contains a boolean field to determine if the switch is digital or analog. There is also a default value field which is used to set the state of the switch when the experiment is started. This field has been given the type of unsigned tiny integer. This allows an 8-bit analog switch to set values to the full resolution of the device. Digital switches may be given the values 0 for off and 255 for on. The experiment ID field is linked to the Experiments table ID field, so that an update on the Experiments table ID field, will update the Switches table. Table 5.11 shows all the information stored in the switches table.

| Field | Type | Required | Primary Key | Unique |
|---|---|---|---|---|
| ID | autonumber | Yes | Yes | Yes |
| Experiment ID | int (unsigned) | Yes | | Yes |
| Description | varchar(255) | Yes | | Yes |
| Default Value | tinyint (unsigned) | Yes | | |
| Is Digital | boolean | Yes | | |

Table 5.11: switches table

### 5.5.8 Booking Table

The booking table is used to book students in for a particular experiment time slot. This table contains the user ID and the experiment ID as well as booking start and finish times. The Username field is linked to the MySQL Membership table Username field and the Experiment ID field is linked to the Experiments table ID field, so that an update on either of these tables will update the Booking table. Table 5.12 shows all the information stored in the booking table.

| Field | Type | Required | Primary Key | Unique |
|---|---|---|---|---|
| UserName | varchar(255) | Yes | Yes | Yes |
| Experiment ID | int (unsigned) | Yes | | |
| Start | Date/Time | Yes | Yes | Yes |
| Finish | Date/Time | Yes | | |

Table 5.12: booking table

### 5.5.9 Relationship

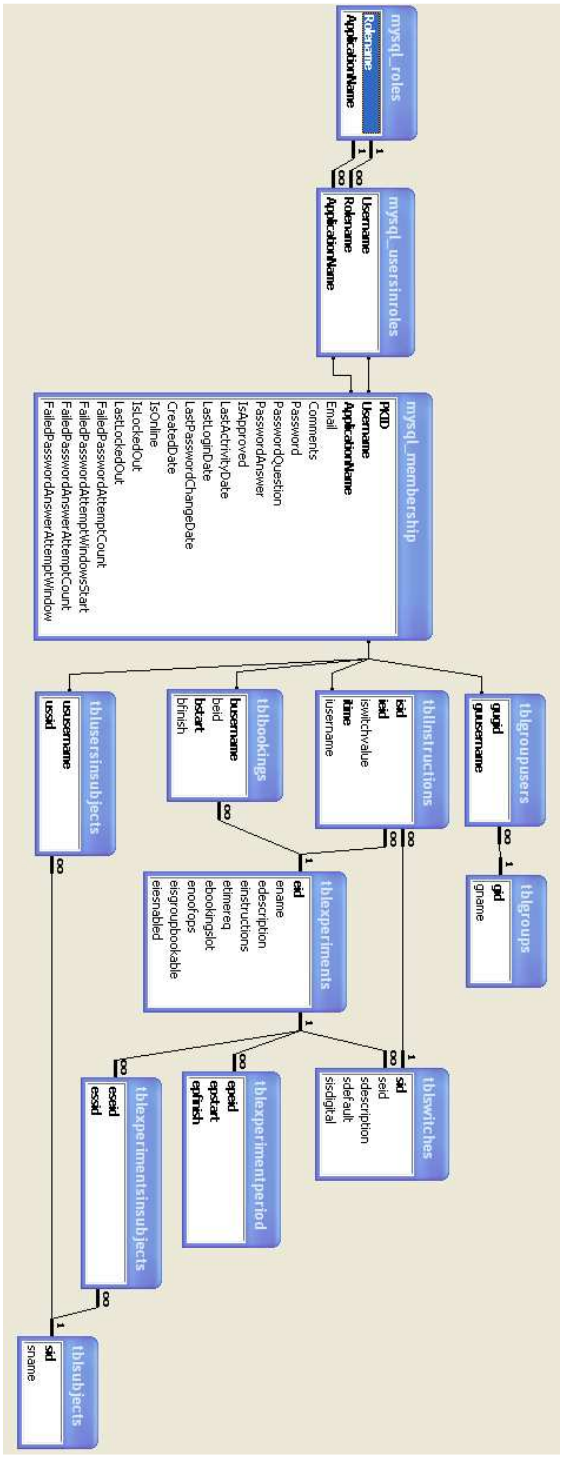


Figure 5.1: Database relationships

# Chapter 6

# Conclusion

Research was conducted into several remote laboratories that are currently in operation. The depth of this research was limited by several factors of which time was the major limiting factor. There are many remote laboratories throughout the world which were not examined because communication with these laboratories would require large amounts of time in language translation or correspondence. Expanding the time allowed for research would possibly yield different alternatives in the structure of a remote laboratory and the technologies used to implement these structures.

Several flaws were identified in the remote laboratories examined, mainly in the area of programming languages used, and measures taken to ensure that these flaws are not transfer to the system being developed. The flaws in these systems were only obtained from a limited knowledge of how the system works, which was obtained by the documentation that was provided and researched. A more comprehensive list of flaws could also have been identified if a team of developers who created and/or maintained a remote laboratory had worked together rather then one persons interpretation.

Research into the software and hardware available was conducted mainly at USQ. Potential consumers were also considered, however it was hoped that by using equipment that was both common and inexpensive, such as common operating systems and by only requiring small amounts of additional hardware or software, potential consumers would be attracted to the system. Equipment costs were reduced by researching several different manufacturers of the equipment required.

Research and evaluation of the tasks required to complete a remote laboratory was conducted and the evaluation was done on the basis of what was capable of running at the university and on the boundaries enforced in this research project.

Programming languages that may be used to complete the tasks to impliment the remote laboratory were researched and evaluation. With so many programming languages being used today only several of the main languages were examined. This in turn limited the scope of this section and as such there may be a better programming language alternatives that were not considered.

A system prototype was partially developed (Code in Appendix C), however the laboratory equipment interface method was too expensive to be purchased and subsequently the Host program could not be developed, leading to the remote laboratory system not being implimenting or testing.

None of the 'As time permits' tasks were completed.

## 6.1 Limitations

By using a MySQL database the limit imposed on the amount of records in a table is 4,294,967,295.

The following limitations have been directly or indirectly created as a result of the size of the fields used in the database and as such may be adjusted in the database before implementation to increase these values. (*MySQL AB* 2007)

| Description | Limit |
|---|---|
| Users in one experiment | 255 |
| Character in User ID | 255 |
| Character in Passwords | 128 |
| Character in First Name | 20 |
| Character in Last Names | 30 |
| Character in Experiment names | 30 |
| Character in Subject codes | 10 |
| Character in Descriptions | 256 |
| Character in Instructions | 65535 |
| Character in Group Names | 20 |
| Character in Switch Descriptions | 256 |
| Switch values Range | 0 to 256 |

Table 6.1: Database Field Limits

## 6.2 Future Work

In the future more detailed research needs to be conducted into the current laboratory system configurations, flaws, and programming languages. This would best be done with a specific research and development team consisting of the creators, operators, and maintainers of any existing remote laboratory system. In addition to this a team to examine the requirements of other potential consumers would provide the system with the flexibility to be integrated into the market outside of the educational and scientific areas.

The system could also be expanded to allow student to program in events to occur at a particular point in time. This can be implemented with a slight change to the operation of the host program and the interface program as the database already excepts the time field in the instructions table. Further investigation could be conducted into how to provide the user with visual and audio feedback through a website without the delay that was experienced in this research project.

# References

*180 Degree, 5 or 3 Finger Robot Arm* (2007).
http://www.mrisar.com/robot-arm-5-f-180.htm
current June 2007.

*Asus Motherboard* (2007).
http://au.asus.com/products1.aspx?l1=3
current June 2007.

Axelson, J. (2005), *USB Complete*, 3rd edn, Lakeview Reasearch LLC.

*Benefits and Constraints of using Microsoft Access Database & Office* (2007).
http://www.galleryimage.com.au/Why-Access-Database.htm
current July 2007.

*Choosing The Right Programming Language* (2007).
http://www.dmxzone.com/ShowDetail.asp?NewsId=4305
current July 2007.

*C - Wikipedia* (2007).
http://en.wikipedia.org/wiki/C%2B%2B
current June 2007.

*Data Driven Attacks Using Http Tunneling* (2007).
http://www.securityfocus.com/infocus/1793
current April 2007.

Davidson, P. & Griffin, R. W. (2000), *Management*, 3rd edn, John Wiley and Sons Australia, Ltd.

*Fwink* (2007).

`http://lundie.ca/fwink/`

current April 2007.

*Gigabyte Motherboard* (2007).

`http://www.gigabyte.com.tw/Products/Motherboard/Default.aspx`

current June 2007.

*Hobbytron* (2007).

`http://www.hobbytron.com/RobotArmKits.html`

current June 2007.

*Http- Tunnel* (2007).

`http://www.http-tunnel.com/html/support/faq.asp#16`

current June 2007.

*Mono brings Visual Basic Programs to Linux* (2007).

`http://www.linuxdevices.com/news/NS9725385854.html`

current July 2007.

*MSDE for Microsft Visual Studio 6.0: An Alternative to Jet for Building* (2007).

`http://msdn2.microsoft.com/en-us/library/ms811092.aspx`

current August 2007.

Murray, S. & Lasky, V. (2007), *UTS Engineering Remote Laboratory Project*.

*MySQL 5.0 vs. Microsoft SQL Server 2005* (2007).

`http://www.tometasoftware.com/MySQL-5-vs-Microsoft-SQL-Server-2005.asp`

current July 2007.

*MySQL AB* (2007).

`http://www.mysql.com/`

current September 2007.

*Optocoupler - Wikipedia* (2007).

`http://en.wikipedia.org/wiki/Optocoupler`

current June 2007.

*RealVNC* (2007).

`http://www.vnc.com/download.html`

current June 2007.

*Relay - Wikipedia* (2007).

`http://en.wikipedia.org/wiki/Relay`

current June 2007.

*Robotic Arm System* (2007).

`http://www.roboworld.com.sg/roboshop/product_list.aspx`

current June 2007.

*Servomechanism - Wikipedia* (2007).

`http://en.wikipedia.org/wiki/Servomechanism`

current June 2007.

*SQL Server vs MySQL* (2007).

`http://searchsqlserver.techtarget.com/tip/`

current July 2007.

*SQL Server vs Oracle* (2007).

`http://www.mssqlcity.com/Articles/Compare/sql_server_vs_oracle.htm`

current July 2007.

The University of Southern Queensland (2006*a*), *Civil Materials Practice*, The University of Southern Queensland.

The University of Southern Queensland (2006*b*), *Computer Systems Engineering Practice*, The University of Southern Queensland.

The University of Southern Queensland (2006*c*), *Electrical and Electronic Practice A*, The University of Southern Queensland.

The University of Southern Queensland (2006*d*), *Electrical and Electronic Practice B*, The University of Southern Queensland.

The University of Southern Queensland (2006*e*), *Electrical and Electronic Practice C*, The University of Southern Queensland.

The University of Southern Queensland (2006*f*), *Electrical and Electronic Practice D*, The University of Southern Queensland.

The University of Southern Queensland (2006*g*), *Engineering Management Introductory BOOK*, The University of Southern Queensland.

The University of Southern Queensland (2006*h*), *Engineering Management Study BOOK 2*, The University of Southern Queensland.

The University of Southern Queensland (2006*i*), *Field Practice*, The University of Southern Queensland.

The University of Southern Queensland (2006*j*), *Mechatronic Practice 1*, The University of Southern Queensland.

The University of Southern Queensland (2006*k*), *Professional Practice 1*, The University of Southern Queensland.

The University of Southern Queensland (2006*l*), *Professional Practice 2*, The University of Southern Queensland.

The University of Southern Queensland (2006*m*), *Soil and Water Engineering Practice 1*, The University of Southern Queensland.

The University of Southern Queensland (2006*n*), *Soil and Water Engineering Practice 2*, The University of Southern Queensland.

*The Telelabs Project* (2007).
`http://telerobot.mech.uwa.edu.au/index.html`
current April 2007.

*Towards Sustainable Engineering Practice: Engineering Frameworks for Sustainability* (1997), Institution of Engineers, Australia, Canberra.

*Transistor - Wikipedia* (2007).
`http://en.wikipedia.org/wiki/Transistor`
current June 2007.

*USB* (2007).
`http://www.webopedia.com/TERM/U/USB.html`
current April 2007.

*Using a Linux L2TP/IPsec VPN server with Mac OS X* (2007).
`http://www.jacco2.dds.nl/networking/freeswan-panther.html`
current June 2007.

*Visual Basic - Wikipedia* (2007).
`http://en.wikipedia.org/wiki/Visual_Basic`
current June 2007.

*What are the capacities of Access, SQL Server, and MSDE?* (2007).
`http://sqlserver2000.databases.aspfaq.com/what-are-...-msde.html`
current July 2007.

Wong, K., Ferguson, C., Florance, J., Bantwal, B. & Jones, T. (2007), *Flexible Delivery of Practical Learning Experience Through the Internet: The Remotely Operated CNC Machine Teaching Project*.

# Appendix A

# Project Specification

**University of Southern Queensland**
**Faculty of Engineering and Surveying**

**ENG4111/4112 Research Project**
**Project Specification**

**For:** Aaron Birkbeck
**Topic:** Remote Laboratory Software
**Supervisor:** John Leis
**Project Aim:** This project aims to research and develop the software required to implement a remote laboratory at the University of Southern Queensland, with the purpose of allowing students to complete selected laboratory experiments offered by USQ's practical courses without attending residential school.

**Programme:** Issue A, 26th March

1. Research current remote laboratories to discover the basic structure of how a remote laboratory works.
2. Investigate how other programmers have developed their software
3. Identify flaws associated with currently used methods.
4. Research the software and hardware available at USQ and other potential consumers.
5. Identify which platform the system will run on.
6. Determine any additional hardware required for the remote laboratory.
7. Evaluate hardware requirements and recommend a vendor.
8. Research alternative methods for the tasks required to implement the remote laboratory software.
9. Evaluate the alternative methods of performing those tasks.
10. Explore potential programming languages that could be used to perform the tasks required.
11. Assess programming languages to determine which best implements the remote laboratory, and can operate on the hardware and software used by the consumers.
12. Develop a prototype of the remote laboratory software.

**As time Permits:**

13. Extensively test the system to determine flaws.
14. Gain user feedback on the operation of the system.
15. Implement changes to correct any flaws identified from testing and feedback.

**Agreed:**

ABirkbeck (Student) _____ (Supervisor)

15/05/2007 17/5/07

# Appendix B

# MySQL Database Code

```
create database RLS;

use RLS;

create user InetUser;
Grant all on RLS.* to InetUser;
Set Password for InetUser = Password('RLSUser');

create table tblgroups(
gid int unsigned not null auto_increment default NULL,
gname varchar(30) not null,
PRIMARY KEY(gid)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci;

create unique index tblgroupsI on tblgroups (gname);

create table tblsubjects(
sid  varchar(10) not null,
sname varchar(255) not null,
PRIMARY KEY(sid, sname)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci;

create unique index tblsubjectsI on tblsubjects (sname);
```

```
create table tblswitches(
sid int unsigned not null auto_increment default NULL,
seid int unsigned not null,
sdescription varchar(256) not null,
sdefaultvalue tinyint unsigned not null,
sisdigital boolean not null,
PRIMARY KEY(sid)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci;

create unique index tblewithcesi on tblSwitches (seid, sdescription);

create table tblexperiments(
eid int unsigned not null auto_increment default NULL,
ename varchar(30) not null,
edescription varchar(256),
einstructions text,
etimereq time not null,
ebookingslot time not null,
enoofops tinyint unsigned not null,
egroupbookable boolean not null,
eenabled boolean not null,
PRIMARY KEY(eid)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci;

create unique index tblexpi on tblexperiments (ename, esubjectcode);
```

```
create table tblgroupusers(
guusername varchar(255) not null,
gugid int unsigned not null,
FOREIGN KEY (guusername) REFERENCES mysql_membership (username)
    ON UPDATE CASCADE ON DELETE RESTRICT,
FOREIGN KEY (gugid) REFERENCES tblgroups(gid)
    ON UPDATE CASCADE ON DELETE RESTRICT,
PRIMARY KEY(guusername, gugid)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci;

create table tblusersinsubjects(
ususername varchar(255) not null,
ussid  varchar(10) not null,
FOREIGN KEY (ususername) REFERENCES mysql_membership (username)
    ON UPDATE CASCADE ON DELETE RESTRICT,
FOREIGN KEY (ussid) REFERENCES tblsubjects(sid)
    ON UPDATE CASCADE ON DELETE RESTRICT,
PRIMARY KEY(ususername, ussid)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci;

create table tblexperimentsinsubjects(
eseid int unsigned not null
essid  varchar(10) not null,
FOREIGN KEY (eseid) REFERENCES tblexperiments (eid)
    ON UPDATE CASCADE ON DELETE RESTRICT,
FOREIGN KEY (ussid) REFERENCES tblsubjects(sid)
    ON UPDATE CASCADE ON DELETE RESTRICT,
PRIMARY KEY(ususername, ussid)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci;
```

```
create table tblexperimentperiod(
epeid int unsigned not null,
epstart datetime not null,
epfinish datetime not null,
FOREIGN KEY (epeid) REFERENCES tblexperiments(eid)
    ON UPDATE CASCADE ON DELETE RESTRICT,
PRIMARY KEY(epeid,epstart,epfinish)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci;

create table tblbookings(
busername varchar(255) not null,
beid int unsigned not null,
bstart datetime not null,
bfinish datetime not null,
FOREIGN KEY (busername) REFERENCES mysql_membership (username)
    ON UPDATE CASCADE ON DELETE RESTRICT,
FOREIGN KEY (beid) REFERENCES tblexperiments(eid)
    ON UPDATE CASCADE ON DELETE RESTRICT,
PRIMARY KEY(busername, bstart)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci;
```

```
create table tblinstructions(
isid int unsigned not null ,
ieid int unsigned not null,
iswitchvalue varchar(256) not null,
itime tinyint unsigned not null,
iusername boolean not null,
FOREIGN KEY (isid) REFERENCES tblswitches (sid)
    ON UPDATE CASCADE ON DELETE RESTRICT,
FOREIGN KEY (ieid) REFERENCES tblexperiments(eid)
    ON UPDATE CASCADE ON DELETE RESTRICT,
FOREIGN KEY (iusername) REFERENCES mysql_membership (username)
    ON UPDATE CASCADE ON DELETE RESTRICT,
PRIMARY KEY(isid, ieid, itime)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci;
```

# Appendix C

# Interface Program Code

## C.1 Index.ASPX

```
<%@ Page Language="vb" AutoEventWireup="false" CodeBehind="index.aspx.vb"
    Inherits="RemoteLab.index" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
    <head runat="server">
        <title>USQ Remote Laboratory</title>
        <link href="styles/default.css" rel="stylesheet" type="text/css" />
    </head>
    <body>
        <form id="form1" runat="server" style="background-color:white">
            <center>
                <br />
                <a href="/student/mainmenu.aspx">Enter</a>
                <br /><br />
            </center>
        </form>
    </body>
</html>
```

## C.2 Index.ASPX.VB

```
Public Partial Class index
    Inherits System.Web.UI.Page
End Class
```

## C.3 Login.ASPX

```
<%@ Page Language="vb" AutoEventWireup="false" CodeBehind="login.aspx.vb"
    Inherits="RemoteLab.login" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
    <head id="Head1" runat="server">
        <title>USQ Remote Laboratory Login</title>
        <link href="styles/default.css" rel="stylesheet" type="text/css" />
    </head>
    <body>
        <form id="form1" runat="server" style="background-color:white">
            <asp:ScriptManager ID="ScriptManager1" runat="server" />
            <asp:UpdatePanel ID="UpdatePanel1" runat="server">
                <ContentTemplate>
                    <center>
                        <br />
                        <asp:Login ID="Login1" runat="server" DestinationPageUrl=
                            "/student/mainmenu.aspx" BackColor="#F7F6F3"
                            BorderColor="#E6E2D8" BorderStyle="Solid"
                            BorderWidth="1px" Font-Names="Verdana"
                            Font-Size="0.8em" PasswordRecoveryText=
                            "Forgot your Password?" PasswordRecoveryUrl=
                            "~/passwordrecovery.aspx" BorderPadding="4"
                            ForeColor="#333333" >
                            <TitleTextStyle BackColor="#5D7B9D" Font-Bold="True"
                                ForeColor="White" Font-Size="0.9em" />
                            <InstructionTextStyle Font-Italic="True"
                                ForeColor="Black" />
                            <TextBoxStyle Font-Size="0.8em" />
```

```
                            <LoginButtonStyle BackColor="#FFFBFF"
                                BorderColor="#CCCCCC" BorderStyle="Solid"
                                BorderWidth="1px" Font-Names="Verdana"
                                Font-Size="0.8em" ForeColor="#284775" />
                        </asp:Login>
                        <br /><br />
                    </center>
                </ContentTemplate>
            </asp:UpdatePanel>
        </form>
    </body>
</html>
```

## C.4 Login.ASPX.VB

```
Public Partial Class login
    Inherits System.Web.UI.Page
End Class
```

## C.5 Passwordrecovery.ASPX

```
<%@ Page Language="vb" AutoEventWireup="false" MasterPageFile="~/default.Master"
    CodeBehind="passwordrecovery.aspx.vb" Inherits="RemoteLab.passwordrecovery"
    title="Untitled Page" %>
<asp:Content ID="Content1" ContentPlaceHolderID="cphMain" runat="server">
    <asp:ScriptManager ID="ScriptManager1" runat="server"></asp:ScriptManager>
    <asp:UpdatePanel ID="UpdatePanel1" runat="server">
        <ContentTemplate>
            <center>
                <asp:PasswordRecovery ID="PasswordRecovery1" runat="server"
                    BackColor="#F7F6F3" BorderColor="#E6E2D8" BorderPadding="4"
                    BorderStyle="Solid" BorderWidth="1px" Font-Names="Verdana"
                    Font-Size="0.8em">
                    <InstructionTextStyle Font-Italic="True" ForeColor="Black" />
                    <SuccessTextStyle Font-Bold="True" ForeColor="#5D7B9D" />
                    <TextBoxStyle Font-Size="0.8em" />
                    <TitleTextStyle BackColor="#5D7B9D" Font-Bold="True"
                        Font-Size="0.9em" ForeColor="White" />
                    <SubmitButtonStyle BackColor="#FFFBFF" BorderColor="#CCCCCC"
                        BorderStyle="Solid" BorderWidth="1px" Font-Names="Verdana"
                        Font-Size="0.8em" ForeColor="#284775" />
                </asp:PasswordRecovery>
            </center>
        </ContentTemplate>
    </asp:UpdatePanel>
</asp:Content>
```

## C.6 Passwordrecovery.ASPX.VB

```
Public Partial Class passwordrecovery
    Inherits System.Web.UI.Page
End Class
```

## C.7 Mainmenu.ASPX

```
<%@ Page Language="vb" AutoEventWireup="false" MasterPageFile="~/default.master"
    CodeBehind="mainmenu.aspx.vb" Inherits="RemoteLab.mainmenu"
    title="Main Menu" %>
<asp:Content ID="Content1" ContentPlaceHolderID="cphmain" Runat="Server">
    <table class="tblMain">
        <tr>
            <td>
                <div id="idAdminSection" runat="server">
                    <asp:Table ID="Table1" runat="server"
                        CssClass="tblCenteredWide">
                        <asp:TableHeaderRow>
                            <asp:TableHeaderCell ColumnSpan="3">
                                Administratation</asp:TableHeaderCell>
                        </asp:TableHeaderRow>
                        <asp:TableRow>
                            <asp:TableCell><a href="../admin/editsubjects.aspx">
                                Add\Edit Subjects</a></asp:TableCell>
                            <asp:TableCell></asp:TableCell>
                            <asp:TableCell><a href="/admin/editgroups.aspx">
                                Add\Edit Groups</a></asp:TableCell>
                        </asp:TableRow>
                        <asp:TableRow>
                            <asp:TableCell>Add\Edit Users to Groups</asp:TableCell>
                            <asp:TableCell><a href="/admin/editusers.aspx">
```

```
                            Add\Edit Users</a></asp:TableCell>
                        <asp:TableCell>Add\Edit Users to Subjects
                            </asp:TableCell>
                    </asp:TableRow>
                    <asp:TableRow>
                        <asp:TableCell>Add\Edit Experiment Switches
                            </asp:TableCell>
                        <asp:TableCell>Add\Edit Experiments
                            </asp:TableCell>
                        <asp:TableCell>Add\Edit Experiment Periods
                            </asp:TableCell>
                    </asp:TableRow>
                </asp:Table>
            </div>
        </td>
    </tr>
    <tr>
        <td>
            <div id="idAccountSettings" runat="server">
                <asp:Table ID="Table2" runat="server" Width="100%"
                    CssClass="tblCenteredWide">
                    <asp:TableHeaderRow>
                        <asp:TableHeaderCell>Account Settings
                            </asp:TableHeaderCell>
                    </asp:TableHeaderRow>
                    <asp:TableRow>
                        <asp:TableCell><a href="/student/changepassword.aspx">
                            Change Password</a></asp:TableCell>
                    </asp:TableRow>
                </asp:Table>
            </div>
        </td>
    </tr>
```

```
        <tr>
            <td>
                <div id="idCurrentExperiments" runat="server">
                    <asp:Table ID="Table3" runat="server" Width="100%"
                        CssClass="tblCenteredWide">
                        <asp:TableHeaderRow>
                            <asp:TableHeaderCell>Current Experiments
                                </asp:TableHeaderCell>
                        </asp:TableHeaderRow>
                    </asp:Table>
                </div>
            </td>
        </tr>
        <tr>
            <td>
                <div id="idExperimentInfo" runat="server">
                    <asp:Table ID="Table4" runat="server" Width="100%"
                        CssClass="tblCenteredWide">
                        <asp:TableHeaderRow>
                            <asp:TableHeaderCell>Experiment Information
                                </asp:TableHeaderCell>
                        </asp:TableHeaderRow>
                    </asp:Table>
                </div>
            </td>
        </tr>
    </table>
</asp:Content>
```

## C.8 Mainmenu.ASPX.VB

```
Public Partial Class mainmenu
    Inherits System.Web.UI.Page


    Protected Sub Page_Load(ByVal sender As Object,
        ByVal e As System.EventArgs) Handles Me.Load
        If Roles.IsUserInRole(HttpContext.Current.User.Identity.Name,
            "Admins") = True Then
            idAdminSection.Visible = True
            idAccountSettings.Visible = True
            idCurrentExperiments.Visible = True
            idExperimentInfo.visible = True
        Else
            idAdminSection.Visible = False
            idAccountSettings.Visible = True
            idCurrentExperiments.Visible = True
            idExperimentInfo.Visible = True
        End If
    End Sub
End Class
```

## C.9 Changepassword.ASPX

```
<%@ Page Language="vb" AutoEventWireup="false" MasterPageFile="~/default.Master"
    CodeBehind="changepassword.aspx.vb" Inherits="RemoteLab.changepassword"
    title="Untitled Page" %>
<asp:Content ID="Content1" ContentPlaceHolderID="cphMain" runat="server">
    <asp:ScriptManager ID="ScriptManager1" runat="server"></asp:ScriptManager>
    <asp:UpdatePanel ID="UpdatePanel1" runat="server">
        <ContentTemplate>
            <center>
                <asp:ChangePassword ID="ChangePassword1" runat="server"
                    BackColor="#F7F6F3" BorderColor="#E6E2D8"
                    BorderPadding="4" BorderStyle="Solid" BorderWidth="1px"
                    Font-Names="Verdana" Font-Size="0.8em"
                    CancelDestinationPageUrl="/student/mainmenu.aspx">
                    <CancelButtonStyle BackColor="#FFFBFF"
                        BorderColor="#CCCCCC" BorderStyle="Solid"
                        BorderWidth="1px" Font-Names="Verdana"
                        Font-Size="0.8em" ForeColor="#284775" />
                    <InstructionTextStyle Font-Italic="True" ForeColor="Black" />
                    <PasswordHintStyle Font-Italic="True" ForeColor="#888888" />
                    <ChangePasswordButtonStyle BackColor="#FFFBFF"
                        BorderColor="#CCCCCC" BorderStyle="Solid"
                        BorderWidth="1px" Font-Names="Verdana"
                        Font-Size="0.8em" ForeColor="#284775" />
                    <ContinueButtonStyle BackColor="#FFFBFF"
                        BorderColor="#CCCCCC" BorderStyle="Solid"
                        BorderWidth="1px" Font-Names="Verdana"
                        Font-Size="0.8em" ForeColor="#284775" />
                    <TitleTextStyle BackColor="#5D7B9D"
                        Font-Bold="True" Font-Size="0.9em" ForeColor="White" />
                    <TextBoxStyle Font-Size="0.8em" />
                </asp:ChangePassword>
```

```
            </center>
        </ContentTemplate>
    </asp:UpdatePanel>
</asp:Content>
```

## C.10 Changepassword.ASPX.VB

```
Public Partial Class changepassword
    Inherits System.Web.UI.Page
End Class
```

## C.11 Editgroup.ASPX

```
<%@ Page Language="vb" AutoEventWireup="false" MasterPageFile="~/default.master"
    CodeBehind="editgroups.aspx.vb" Inherits="RemoteLab.editgroups"
    title="Untitled Page" %>
<asp:Content ID="Content1" ContentPlaceHolderID="cphmain" Runat="Server">
    <asp:ScriptManager ID="ScriptManager1" runat="server"></asp:ScriptManager>
    <asp:UpdatePanel ID="UpdatePanel1" runat="server">
        <ContentTemplate>
            <center>
                <h2>Groups</h2>
                <table id="tblAddInfo" runat="server" class="tblAddNew">
                    <tr>
                        <th>Group Name:</th>
                        <td></td>
                    </tr>
                    <tr>
                        <td><asp:TextBox ID="txtAddGroup" Runat="server" />
                            <asp:RequiredFieldValidator
```

```
                    Runat="server"
                    ID="Requiredfieldvalidator1"
                    EnableClientScript="False"
                    SetFocusOnError = "true"
                    Display="Dynamic"
                    ValidationGroup="Add"
                    controlToValidate="txtAddGroup"
                    ErrorMessage="Group Name."
                    >*</asp:RequiredFieldValidator>
                <asp:RegularExpressionValidator
                    Runat="server"
                    ID="Regularexpressionvalidator1"
                    EnableClientScript="False"
                    SetFocusOnError = "true"
                    Display="Dynamic"
                    ValidationGroup="Add"
                    controlToValidate="txtAddGroup"
                    ValidationExpression="[\w\s \-]{1,20}"
                    ErrorMessage="Group Name."
                    >*</asp:RegularExpressionValidator>
            </td>
            <td><asp:LinkButton ID="IDAdd" runat="server" Text="Add"
                OnClick="IDAdd_Click" /></td>
        </tr>
    </table>
    <asp:ValidationSummary ID="ValidationSummary1"
            DisplayMode="SingleParagraph" ShowSummary="True"
            ValidationGroup="Add" Runat="server" HeaderText=
            "You must enter a valid input in the following fields:"/>
    <br/>
    <asp:DataGrid ID="dgdGroup" runat="server"
            HorizontalAlign="Center" CssClass="DGNormal"
            AutoGenerateColumns="False" DataKeyField="gid"
```

```
                    OnEditCommand="dgdGroup_EditCommand"
                    OnUpdateCommand="dgdGroup_UpdateCommand"
                    OnCancelCommand="dgdGroup_CancelCommand"
                    OnItemCommand="dgdGroup_ItemCommand">
                <Columns>
                    <asp:TemplateColumn HeaderStyle-Width="30"
                        HeaderText="Group ID:" HeaderStyle-CssClass=
                        "DGHNormal" Visible="false">
                        <ItemTemplate>
                            <asp:Literal ID="litEditGroupID" Text=
                                '<%#DataBinder.Eval(Container.DataItem,
                                "gid")%>' Runat="server" />
                        </ItemTemplate>
                    </asp:TemplateColumn>
                    <asp:TemplateColumn HeaderStyle-Width="100px"
                        HeaderText="Group Name:" HeaderStyle-CssClass=
                        "DGHNormal">
                        <ItemTemplate>
                            <asp:Label ID="lblEditGroupName" Text=
                                '<%#DataBinder.Eval(Container.DataItem,
                                "gname")%>' Runat="server"/>
                        </ItemTemplate>
                        <EditItemTemplate>
                            <asp:TextBox ID="txtEditGroup" Text=
                                '<%#DataBinder.Eval(Container.DataItem,
"gname")%>' Runat="server"/>
                            <asp:RequiredFieldValidator
                                Runat="server"
                                ID="Requiredfieldvalidator1"
                                EnableClientScript="False"
                                SetFocusOnError = "true"
                                Display="Dynamic"
                                ValidationGroup="Edit"
```

```
                                    controlToValidate="txtEditGroup"
                                    ErrorMessage="Group Name."
                                    >*</asp:RequiredFieldValidator>
                                <asp:RegularExpressionValidator
                                    Runat="server"
                                    ID="Regularexpressionvalidator1"
                                    SetFocusOnError = "true"
                                    EnableClientScript="False"
                                    Display="Dynamic"
                                    ValidationGroup="Edit"
                                    controlToValidate="txtEditGroup"
                                    ValidationExpression="[\w\s \-]{1,20}"
                                    ErrorMessage="Group Name."
                                    >*</asp:RegularExpressionValidator>
                            </EditItemTemplate>
                        </asp:TemplateColumn>
                        <asp:EditCommandColumn EditText="Edit" UpdateText=
                            "Update" CancelText="Cancel" />
                        <asp:ButtonColumn CommandName="Delete" Text="Delete" />
                    </Columns>
                </asp:DataGrid>
                <asp:ValidationSummary ID="ValidationSummary2" DisplayMode=
                    "SingleParagraph" ShowSummary="True" ValidationGroup="Edit"
                    Runat="server" HeaderText=
                    "You must enter a valid input in the following fields:"/>
                <asp:Label ID="lblDBErrors" runat="server"
                    CssClass="DBErrors"/>
            </center>
        </ContentTemplate>
    </asp:UpdatePanel>
</asp:Content>
```

## C.12 Editgroup.ASPX.VB

```
Imports RemoteLab.MySQLConnection
Partial Public Class editgroups
    Inherits System.Web.UI.Page
    Protected Overrides Sub OnInit(ByVal e As System.EventArgs)
        If (Not IsPostBack) Then
            BindDataGrid()
        End If
    End Sub
    Protected Sub IDAdd_Click(ByVal sender As System.Object,
        ByVal e As System.EventArgs)
        ' Clear Errors
        ClearDBErrors()
        Page.Validate("Add")
        If Page.IsValid Then
            Try
                Dim GroupQuery As New MySQLQuery(QueryNums.InsertGroup, "",
                txtAddGroup.Text.Trim)
                BindDataGrid()
            Catch ex As MySQLException
                ShowDBErrors(ex)
            End Try
        End If
    End Sub
    Sub dgdGroup_EditCommand(ByVal sender As Object,
        ByVal e As DataGridCommandEventArgs)
        tblAddInfo.Visible = False
        ClearDBErrors()
        dgdGroup.EditItemIndex = e.Item.ItemIndex
        BindDataGrid()
    End Sub
    Sub dgdGroup_UpdateCommand(ByVal sender As Object,
```

```
        ByVal e As DataGridCommandEventArgs)
        Dim litGroupID As Literal = e.Item.FindControl("litEditGroupID")
        Dim txtGroupName As TextBox = e.Item.FindControl("txtEditGroup")
        ClearDBErrors()
        Page.Validate("Edit")
        If Page.IsValid Then
            Try
                Dim GroupsQuery As New MySQLQuery(QueryNums.AppendGroup,
                    "", txtGroupName.Text.Trim, litGroupID.Text.Trim)
            Catch ex As MySQLException
                ShowDBErrors(ex)
            End Try
            tblAddInfo.Visible = True
            dgdGroup.EditItemIndex = -1
            BindDataGrid()
        End If
    End Sub
    Sub dgdGroup_ItemCommand(ByVal sender As Object,
        ByVal e As DataGridCommandEventArgs)
        Select Case e.CommandName
            Case "Delete"
                Dim litGroupID As Literal = e.Item.FindControl("litEditGroupID")
                Try
                    Dim GroupsQuery As New MySQLQuery(QueryNums.DeleteGroup, "",
                        litGroupID.Text.Trim)
                Catch ex As MySQLException
                    ShowDBErrors(ex)
                End Try
                BindDataGrid()
                If dgdGroup.Items.Count < 1 Then
                    CancelAction()
                End If
            Case Else
```

```
        End Select
    End Sub
    Sub dgdGroup_CancelCommand(ByVal sender As Object,
        ByVal e As DataGridCommandEventArgs)
        CancelAction()
    End Sub
    Private Sub BindDataGrid()
        Try
            Dim GroupsQuery As New MySQLQuery(QueryNums.SelectGroups,
                "ORDER BY gname")
            dgdGroup.DataSource = GroupsQuery.Reader
            dgdGroup.DataBind()
        Catch ex As Exception
        End Try
    End Sub
    Private Sub CancelAction()
        tblAddInfo.Visible = True
        dgdGroup.EditItemIndex = -1
        BindDataGrid()
    End Sub
    Private Sub ClearDBErrors()
        lblDBErrors.Text = ""
    End Sub
    Private Sub ShowDBErrors(ByVal ex As MySQLException)
        lblDBErrors.Text = ex.message
    End Sub
End Class
```

## C.13 Editsubjects.ASPX

```
<%@ Page Language="vb" AutoEventWireup="false" MasterPageFile="~/default.Master"
    CodeBehind="editsubjects.aspx.vb" Inherits="RemoteLab.editsubjects"
    title="Untitled Page" %>
<asp:Content ID="Content1" ContentPlaceHolderID="cphMain" runat="server">
    <asp:ScriptManager ID="ScriptManager1" runat="server"></asp:ScriptManager>
    <asp:UpdatePanel ID="UpdatePanel1" runat="server">
        <ContentTemplate>
            <center>
                <h2>Subjects</h2>
                <table id="tblAddInfo" runat="server" class="tblAddNew">
                    <tr>
                        <th>Code:</th>
                        <th>Name:</th>
                        <td></td>
                    </tr>
                    <tr>
                        <td>
                            <asp:TextBox ID="txtAddSubjectCode" Runat="server" />
                            <asp:RequiredFieldValidator
                                Runat="server"
                                ID="Requiredfieldvalidator1"
                                EnableClientScript="False"
                                SetFocusOnError = "true"
                                Display="Dynamic"
                                ValidationGroup="Add"
                                controlToValidate="txtAddSubjectCode"
                                ErrorMessage="Code."
                                >*</asp:RequiredFieldValidator>
                            <asp:RegularExpressionValidator
                                Runat="server"
                                ID="Regularexpressionvalidator1"
```

```
                    EnableClientScript="False"
                    SetFocusOnError = "true"
                    Display="Dynamic"
                    ValidationGroup="Add"
                    controlToValidate="txtAddSubjectCode"
                    ValidationExpression="[\w\s]{1,10}"
                    ErrorMessage="Code."
                    >*</asp:RegularExpressionValidator>
                </td>
                <td>
                    <asp:TextBox ID="txtAddSubjectName" Runat="server" />
                    <asp:RequiredFieldValidator
                        Runat="server"
                        ID="Requiredfieldvalidator2"
                        EnableClientScript="False"
                        SetFocusOnError = "true"
                        Display="Dynamic"
                        ValidationGroup="Add"
                        controlToValidate="txtAddSubjectName"
                        ErrorMessage="Name."
                        >*</asp:RequiredFieldValidator>
                    <asp:RegularExpressionValidator
                        Runat="server"
                        ID="Regularexpressionvalidator2"
                        EnableClientScript="False"
                        SetFocusOnError = "true"
                        Display="Dynamic"
                        ValidationGroup="Add"
                        controlToValidate="txtAddSubjectName"
                        ValidationExpression="[\w\s]{1,255}"
                        ErrorMessage="Name."
                        >*</asp:RegularExpressionValidator>
                </td>
```

```
            <td><asp:LinkButton ID="IDAdd" runat="server" Text="Add"
                OnClick="IDAdd_Click" /></td>
        </tr>
    </table>
    <asp:ValidationSummary ID="ValidationSummary1"
        DisplayMode="SingleParagraph" ShowSummary="True"
        ValidationGroup="Add" Runat="server" HeaderText=
        "You must enter a valid input in the following fields:"/>
    <br/>
    <asp:DataGrid ID="dgdSubject" runat="server"
            HorizontalAlign="Center" CssClass="DGNormal"
            AutoGenerateColumns="False" DataKeyField="sid"
            OnEditCommand="dgdSubject_EditCommand"
            OnUpdateCommand="dgdSubject_UpdateCommand"
            OnCancelCommand="dgdSubject_CancelCommand"
            OnItemCommand="dgdSubject_ItemCommand">
        <Columns>
            <asp:TemplateColumn HeaderStyle-Width="30"
                HeaderText="Code:" HeaderStyle-CssClass="DGHNormal">
                <ItemTemplate>
                    <asp:Literal ID="litEditSubjectCode" Text=
                    '<%#DataBinder.Eval(Container.DataItem, "sid")%>'
                    Runat="server" />
                </ItemTemplate>
            </asp:TemplateColumn>
            <asp:TemplateColumn HeaderStyle-Width="100px"
                HeaderText="Name:" HeaderStyle-CssClass="DGHNormal">
                <ItemTemplate>
                    <asp:Label ID="lblEditSubjectName"
                    Text='<%#DataBinder.Eval(Container.DataItem,
                    "sname")%>' Runat="server"/>
                </ItemTemplate>
                <EditItemTemplate>
```

```
                        <asp:TextBox ID="txtEditSubjectName" Text=
                            '<%#DataBinder.Eval(Container.DataItem,
                            "sname")%>' Runat="server"/>
                        <asp:RequiredFieldValidator
                            Runat="server"
                            ID="Requiredfieldvalidator4"
                            EnableClientScript="False"
                            SetFocusOnError = "true"
                            Display="Dynamic"
                            ValidationGroup="Edit"
                            controlToValidate="txtEditSubjectName"
                            ErrorMessage="Name."
                            >*</asp:RequiredFieldValidator>
                        <asp:RegularExpressionValidator
                            Runat="server"
                            ID="Regularexpressionvalidator4"
                            SetFocusOnError = "true"
                            EnableClientScript="False"
                            Display="Dynamic"
                            ValidationGroup="Edit"
                            controlToValidate="txtEditSubjectName"
                            ValidationExpression="[\w\s]{1,255}"
                            ErrorMessage="Name."
                            >*</asp:RegularExpressionValidator>
                    </EditItemTemplate>
                </asp:TemplateColumn>
                <asp:EditCommandColumn EditText="Edit" UpdateText="Update"
                    CancelText="Cancel" />
                <asp:ButtonColumn CommandName="Delete" Text="Delete" />
            </Columns>
        </asp:DataGrid>
        <asp:ValidationSummary ID="ValidationSummary2"
            DisplayMode="SingleParagraph" ShowSummary="True"
```

```
                    ValidationGroup="Edit" Runat="server" HeaderText=
                    "You must enter a valid input in the following fields:"/>
                <asp:Label ID="lblDBErrors" runat="server"
                    CssClass="DBErrors"/>
            </center>
        </ContentTemplate>
    </asp:UpdatePanel>
</asp:Content>
```

## C.14 Editsubjects.ASPX.VB

```
Imports RemoteLab.MySQLConnection
Partial Public Class editsubjects
    Inherits System.Web.UI.Page
    Protected Overrides Sub OnInit(ByVal e As System.EventArgs)
        If (Not IsPostBack) Then
            BindDataGrid()
        End If
    End Sub
    Protected Sub IDAdd_Click(ByVal sender As System.Object,
        ByVal e As System.EventArgs)
        ' Clear Errors
        ClearDBErrors()
        Page.Validate("Add")
        If Page.IsValid Then
            Try
                Dim GroupQuery As New MySQLQuery(QueryNums.InsertSubject, "",
                    txtAddSubjectCode.Text.Trim, txtAddSubjectName.Text.Trim)
                BindDataGrid()
            Catch ex As MySQLException
                ShowDBErrors(ex)
            End Try
```

```
        End If
    End Sub
    Sub dgdSubject_EditCommand(ByVal sender As Object,
        ByVal e As DataGridCommandEventArgs)
        tblAddInfo.Visible = False
        ClearDBErrors()
        dgdSubject.EditItemIndex = e.Item.ItemIndex
        BindDataGrid()
    End Sub
    Sub dgdSubject_UpdateCommand(ByVal sender As Object,
        ByVal e As DataGridCommandEventArgs)
        Dim litSubjectCode As Literal = e.Item.FindControl("litEditSubjectCode")
        Dim txtSubjectName As TextBox = e.Item.FindControl("txtEditSubjectName")
        ClearDBErrors()
        Page.Validate("Edit")
        If Page.IsValid Then
            Try
                Dim GroupsQuery As New MySQLQuery(QueryNums.AppendSubject, "",
                   txtSubjectName.Text.Trim, litSubjectCode.Text.Trim)
            Catch ex As MySQLException
                ShowDBErrors(ex)
            End Try
            tblAddInfo.Visible = True
            dgdSubject.EditItemIndex = -1
            BindDataGrid()
        End If
    End Sub
    Sub dgdSubject_ItemCommand(ByVal sender As Object,
        ByVal e As DataGridCommandEventArgs)
        Select Case e.CommandName
            Case "Delete"
                Dim litSubjectCode As Literal =
                    e.Item.FindControl ("litEditSubjectCode")
```

```
            Try
                Dim GroupsQuery As New MySQLQuery(QueryNums.DeleteSubject,
                    "", litSubjectCode.Text.Trim)
            Catch ex As MySQLException
                ShowDBErrors(ex)
            End Try
            BindDataGrid()
            If dgdSubject.Items.Count < 1 Then
                CancelAction()
            End If
        Case Else
    End Select
End Sub
Sub dgdSubject_CancelCommand(ByVal sender As Object,
    ByVal e As DataGridCommandEventArgs)
    CancelAction()
End Sub
Private Sub BindDataGrid()
    Try
        Dim GroupsQuery As New MySQLQuery(QueryNums.SelectSubjects,
            "ORDER BY sid")
        dgdSubject.DataSource = GroupsQuery.Reader
        dgdSubject.DataBind()
    Catch ex As Exception
    End Try
End Sub
Private Sub CancelAction()
    tblAddInfo.Visible = True
    dgdSubject.EditItemIndex = -1
    BindDataGrid()
End Sub
Private Sub ClearDBErrors()
    lblDBErrors.Text = ""
```

```
    End Sub
    Private Sub ShowDBErrors(ByVal ex As MySQLException)
        lblDBErrors.Text = ex.message
    End Sub
End Class
```

## C.15 Editusers.ASPX

```
<%@ Page Language="vb" AutoEventWireup="false" MasterPageFile=
    "~/default.Master" CodeBehind="editusers.aspx.vb"
    Inherits="RemoteLab.editusers" title="Untitled Page" %>
<asp:Content ID="Content1" ContentPlaceHolderID="cphMain"
    runat="server">
    <asp:ScriptManager ID="ScriptManager1" runat="server">
        </asp:ScriptManager>
    <asp:UpdatePanel ID="UpdatePanel1" runat="server">
        <ContentTemplate>
            <center>
                <h2>Users</h2>
                <table id="tblAddInfo" runat="server" class="tblAddNew">
                    <tr>
                        <th>User Name:</th>
                        <td><asp:TextBox ID="txtAddUserName" Runat="server" />
                            <asp:RequiredFieldValidator
                                Runat="server"
                                ID="Requiredfieldvalidator1"
                                EnableClientScript="False"
                                SetFocusOnError = "true"
                                Display="Dynamic"
                                ValidationGroup="Add"
                                controlToValidate="txtAddUserName"
                                ErrorMessage="User Name."
```

```
                    >*</asp:RequiredFieldValidator>
                <asp:RegularExpressionValidator
                    Runat="server"
                    ID="Regularexpressionvalidator1"
                    EnableClientScript="False"
                    SetFocusOnError = "true"
                    Display="Dynamic"
                    ValidationGroup="Add"
                    controlToValidate="txtAddUserName"
                    ValidationExpression="[\w\s \-]{1,255}"
                    ErrorMessage="User Name."
                    >*</asp:RegularExpressionValidator>
            </td>
            <td></td>
        </tr>
        <tr>
            <th>First Name:</th>
            <td><asp:TextBox ID="txtAddFirstName" Runat="server" />
                <asp:RequiredFieldValidator
                    Runat="server"
                    ID="Requiredfieldvalidator2"
                    EnableClientScript="False"
                    SetFocusOnError = "true"
                    Display="Dynamic"
                    ValidationGroup="Add"
                    controlToValidate="txtAddFirstName"
                    ErrorMessage="First Name."
                    >*</asp:RequiredFieldValidator>
                <asp:RegularExpressionValidator
                    Runat="server"
                    ID="Regularexpressionvalidator2"
                    EnableClientScript="False"
                    SetFocusOnError = "true"
```

```
                    Display="Dynamic"
                    ValidationGroup="Add"
                    controlToValidate="txtAddFirstName"
                    ValidationExpression="[\w\s \-]{1,20}"
                    ErrorMessage="First Name."
                    >*</asp:RegularExpressionValidator>
            </td>
            <td></td>
        </tr>
        <tr>
            <th>Last Name:</th>
            <td><asp:TextBox ID="txtAddLastName" Runat="server" />
                <asp:RequiredFieldValidator
                    Runat="server"
                    ID="Requiredfieldvalidator7"
                    EnableClientScript="False"
                    SetFocusOnError = "true"
                    Display="Dynamic"
                    ValidationGroup="Add"
                    controlToValidate="txtAddLastName"
                    ErrorMessage="Last Name."
                    >*</asp:RequiredFieldValidator>
                <asp:RegularExpressionValidator
                    Runat="server"
                    ID="Regularexpressionvalidator3"
                    EnableClientScript="False"
                    SetFocusOnError = "true"
                    Display="Dynamic"
                    ValidationGroup="Add"
                    controlToValidate="txtAddLastName"
                    ValidationExpression="[\w\s \-]{1,20}"
                    ErrorMessage="Last Name."
                    >*</asp:RegularExpressionValidator>
```

```
            </td>
            <td></td>
        </tr>
        <tr>
            <th>Password:</th>
            <td><asp:TextBox ID="txtAddPassword" Runat="server"
                TextMode="Password" />
                <asp:RequiredFieldValidator
                    Runat="server"
                    ID="Requiredfieldvalidator8"
                    EnableClientScript="False"
                    SetFocusOnError = "true"
                    Display="Dynamic"
                    ValidationGroup="Add"
                    controlToValidate="txtAddPassword"
                    ErrorMessage="Password."
                    >*</asp:RequiredFieldValidator>
                <asp:RegularExpressionValidator
                    Runat="server"
                    ID="Regularexpressionvalidator4"
                    EnableClientScript="False"
                    SetFocusOnError = "true"
                    Display="Dynamic"
                    ValidationGroup="Add"
                    controlToValidate="txtAddPassword"
                    ValidationExpression="[\w\s \-]{1,20}"
                    ErrorMessage="Password."
                    >*</asp:RegularExpressionValidator>
            </td>
            <td></td>
        </tr>
        <tr>
            <th>Email:</th>
```

```
                <td><asp:TextBox ID="txtAddEmail" Runat="server" />
                    <asp:RequiredFieldValidator
                        Runat="server"
                        ID="Requiredfieldvalidator9"
                        EnableClientScript="False"
                        SetFocusOnError = "true"
                        Display="Dynamic"
                        ValidationGroup="Add"
                        controlToValidate="txtAddEmail"
                        ErrorMessage="email."
                        >*</asp:RequiredFieldValidator>
                    <asp:RegularExpressionValidator
                        Runat="server"
                        ID="Regularexpressionvalidator5"
                        EnableClientScript="False"
                        SetFocusOnError = "true"
                        Display="Dynamic"
                        ValidationGroup="Add"
                        controlToValidate="txtAddEmail"
                        ValidationExpression=
                        "\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*"
                        ErrorMessage="Email."
                        >*</asp:RegularExpressionValidator>
                </td>
                <td></td>
            </tr>
            <tr>
                <th>Role:</th>
                <td>
                    <asp:RadioButtonList ID="rblRole" runat="server">
                        <asp:ListItem Selected="True">Student
                            </asp:ListItem>
                        <asp:ListItem>Administrator</asp:ListItem>
```

```
                </asp:RadioButtonList>
            </td>
            <td> <asp:LinkButton ID="IDAdd" runat="server"
               Text="Add" OnClick="IDAdd_Click" /></td>
        </tr>
    </table>

    <asp:ValidationSummary ID="ValidationSummary1"
        DisplayMode="SingleParagraph" ShowSummary="True"
        ValidationGroup="Add" Runat="server" HeaderText=
        "You must enter a valid input in the following fields:"/>
    <br />
    <asp:DataGrid ID="dgdUsers" runat="server"
        HorizontalAlign="Center" CssClass="DGNormal"
        AutoGenerateColumns="False" DataKeyField="pkid"
        OnEditCommand="dgdUsers_EditCommand"
        OnUpdateCommand="dgdUsers_UpdateCommand"
        OnCancelCommand="dgdUsers_CancelCommand">
        <Columns>
        <asp:TemplateColumn HeaderStyle-Width="30"
            HeaderText="PKID:" HeaderStyle-CssClass="DGHNormal"
            Visible="false">
            <ItemTemplate>
                <asp:Literal ID="litEditPKID" Text=
                '<%#DataBinder.Eval(Container.DataItem,
                "pkid")%>' Runat="server" />
            </ItemTemplate>
        </asp:TemplateColumn>
        <asp:TemplateColumn HeaderStyle-Width="100px"
            HeaderText="Username:" HeaderStyle-CssClass="DGHNormal">
            <ItemTemplate>
                <asp:Label ID="lblEditUsername" Text=
                    '<%#DataBinder.Eval(Container.DataItem,
```

```
                "username")%>' Runat="server"/>
        </ItemTemplate>
    </asp:TemplateColumn>
    <asp:TemplateColumn HeaderStyle-Width="100px"
        HeaderText="First Name:" HeaderStyle-CssClass="DGHNormal">
        <ItemTemplate>
            <asp:Label ID="lblEditFirstname" Text=
                '<%#GetFirstName(DataBinder.Eval
                (Container.DataItem,"comment"))%>'
                Runat="server"/>
        </ItemTemplate>
        <EditItemTemplate>
            <asp:TextBox ID="txtEditFirstname" Text=
            '<%#GetFirstName(DataBinder.Eval(Container.DataItem,
                "comment"))%>' Runat="server"/>
            <asp:RequiredFieldValidator
                Runat="server"
                ID="Requiredfieldvalidator4"
                EnableClientScript="False"
                SetFocusOnError = "true"
                Display="Dynamic"
                ValidationGroup="Edit"
                controlToValidate="txtEditFirstname"
                ErrorMessage="First Name."
                >*</asp:RequiredFieldValidator>
        </EditItemTemplate>
    </asp:TemplateColumn>
    <asp:TemplateColumn HeaderStyle-Width="100px"
        HeaderText="Last Name:" HeaderStyle-CssClass="DGHNormal">
        <ItemTemplate>
            <asp:Label ID="lblEditLastname" Text=
                '<%#GetLastName(DataBinder.Eval
                (Container.DataItem, "comment"))%>'
```

```
                Runat="server"/>
            </ItemTemplate>
            <EditItemTemplate>
                <asp:TextBox ID="txtEditLastname" Text=
                    '<%#GetLastName(DataBinder.Eval
                    (Container.DataItem, "comment"))%>'
                    Runat="server"/>
                <asp:RequiredFieldValidator
                    Runat="server"
                    ID="Requiredfieldvalidator5"
                    EnableClientScript="False"
                    SetFocusOnError = "true"
                    Display="Dynamic"
                    ValidationGroup="Edit"
                    controlToValidate="txtEditLastname"
                    ErrorMessage="Last Name."
                    >*</asp:RequiredFieldValidator>
            </EditItemTemplate>
        </asp:TemplateColumn>
        <asp:TemplateColumn HeaderStyle-Width="100px"
            HeaderText="Email:" HeaderStyle-CssClass="DGHNormal">
            <ItemTemplate>
                <asp:Label ID="lblEditEmail" Text=
                    '<%#DataBinder.Eval(Container.DataItem,
                    "email")%>' Runat="server"/>
            </ItemTemplate>
            <EditItemTemplate>
                <asp:TextBox ID="txtEditEmail" Text=
                    '<%#DataBinder.Eval(Container.DataItem,
                    "email")%>' Runat="server"/>
                <asp:RequiredFieldValidator
                    Runat="server"
                    ID="Requiredfieldvalidator6"
```

```
                        EnableClientScript="False"
                        SetFocusOnError = "true"
                        Display="Dynamic"
                        ValidationGroup="Edit"
                        controlToValidate="txtEditEmail"
                        ErrorMessage="Email."
                        >*</asp:RequiredFieldValidator>
                </EditItemTemplate>
            </asp:TemplateColumn>
            <asp:TemplateColumn HeaderText="Group:"
                HeaderStyle-CssClass="DGHNormal">
                <ItemTemplate>
                    <%#GetRoleName(DataBinder.Eval
                    (Container.DataItem, "username")) %>
                </ItemTemplate>
            </asp:TemplateColumn>
            <asp:TemplateColumn HeaderText="Locked Out?:
                " HeaderStyle-CssClass="DGHNormal">
                <ItemTemplate>
                    <asp:CheckBox ID="chkAddIsLockedOut" Checked=
                        '<%#DataBinder.Eval(Container.DataItem,
                        "IsLockedOut")%>' runat="server" Enabled="false"/>
                </ItemTemplate>
                <EditItemTemplate>
                    <asp:CheckBox ID="chkEditIsLockedOut" Checked=
                        '<%#DataBinder.Eval(Container.DataItem,
                        "IsLockedOut")%>' runat="server" />
                </EditItemTemplate>
            </asp:TemplateColumn>
            <asp:EditCommandColumn EditText="Edit"
                UpdateText="Update" CancelText="Cancel" />
        </Columns>
    </asp:DataGrid>
```

```
            <asp:ValidationSummary ID="ValidationSummary2"
                DisplayMode="SingleParagraph" ShowSummary="True"
                ValidationGroup="Edit" Runat="server"
                HeaderText=
                "You must enter a valid input in the following fields:"/>
            <asp:Label ID="lblDBErrors" runat="server"
                CssClass="DBErrors"/>
            </center>
        </ContentTemplate>
    </asp:UpdatePanel>
</asp:Content>
```

## C.16 Editusers.ASPX.VB

```
Imports RemoteLab.MySQLConnection
Partial Public Class editusers
    Inherits System.Web.UI.Page
    Public Shared Function GetFirstName(ByVal comments) As String
        Dim result() As String
        Dim strcomments As String = comments
        result = strcomments.Split(", ")
        Try
            Return result(1)
        Catch ex As Exception
            Return ""
        End Try
    End Function
    Public Shared Function GetLastName(ByVal comments) As String
        Dim result() As String
        Dim strcomments As String = comments
        result = strcomments.Split(", ")
        Try
```

```
            Return result(0)
        Catch ex As Exception
            Return ""
        End Try
    End Function

    Protected Overrides Sub OnInit(ByVal e As System.EventArgs)
        If (Not IsPostBack) Then
            BindDataGrid()
        End If
    End Sub
    Sub dgdUsers_EditCommand(ByVal sender As Object,
        ByVal e As DataGridCommandEventArgs)
        ClearDBErrors()
        dgdUsers.EditItemIndex = e.Item.ItemIndex
        BindDataGrid()
    End Sub
    Sub dgdUsers_UpdateCommand(ByVal sender As Object,
        ByVal e As DataGridCommandEventArgs)
        Dim litPKID As Literal = e.Item.FindControl("litEditPKID")
        Dim txtUserName As Label = e.Item.FindControl("lblEditUsername")
        Dim txtFirstName As TextBox = e.Item.FindControl("txtEditFirstname")
        Dim txtLastName As TextBox = e.Item.FindControl("txtEditLastname")
        Dim txtEmail As TextBox = e.Item.FindControl("txtEditEmail")
        Dim chkIsLockedOut As CheckBox = e.Item.FindControl("chkEditIsLockedOut")
        ClearDBErrors()
        Page.Validate("Edit")
        If Page.IsValid Then
            Try
                Dim GroupsQuery As New MySQLQuery(QueryNums.AppendUser, "",
                    txtUserName.Text.Trim, _
                    txtLastName.Text.Trim & ", " & txtFirstName.Text.Trim,
                        txtEmail.Text.Trim, chkIsLockedOut.Checked,
```

```
                    litPKID.Text.Trim)
            Catch ex As MySQLException
                ShowDBErrors(ex)
            End Try
            dgdUsers.EditItemIndex = -1
            BindDataGrid()
        End If
    End Sub
    Sub dgdUsers_CancelCommand(ByVal sender As Object,
        ByVal e As DataGridCommandEventArgs)
        CancelAction()
    End Sub
    Private Sub BindDataGrid()
        Try
            Dim GroupsQuery As New MySQLQuery(QueryNums.SelectUsers,
                "ORDER BY comment")
            dgdUsers.DataSource = GroupsQuery.Reader
            dgdUsers.DataBind()
        Catch ex As Exception
        End Try
    End Sub
    Private Sub CancelAction()
        dgdUsers.EditItemIndex = -1
        BindDataGrid()
    End Sub
    Private Sub ClearDBErrors()
        lblDBErrors.Text = ""
    End Sub
    Private Sub ShowDBErrors(ByVal ex As MySQLException)
        lblDBErrors.Text = ex.message
    End Sub
    Shared Function GetRoleName(ByVal Username) As String
        Dim UserRoles As String() = Roles.GetRolesForUser(Username)
```

```
        Return UserRoles(0)
    End Function
    Shared Function IsAdmin(ByVal Username) As String
        If GetRoleName(Username) = "Admins" Then
            Return True
        Else
            Return False
        End If
    End Function
    Shared Function IsStudent(ByVal Username) As String
        If GetRoleName(Username) = "Students" Then
            Return True
        Else
            Return False
        End If
    End Function

    Protected Sub IDAdd_Click(ByVal sender As System.Object,
        ByVal e As System.EventArgs)
        Dim user As MembershipUser
        Dim status As MembershipCreateStatus
        Membership.CreateUser(txtAddUserName.Text.Trim,
            txtAddPassword.Text.Trim, txtAddEmail.Text.Trim,
            "Student Number", txtAddUserName.Text.Trim,
            True, status)
        If status = MembershipCreateStatus.Success Then
            user = Membership.GetUser(txtAddUserName.Text.Trim)
            user.Comment = txtAddFirstName.Text.Trim & " " &
                txtAddLastName.Text.Trim
            If rblRole.SelectedItem.Text = "Administrator" Then
                Roles.AddUserToRole(txtAddUserName.Text.Trim, "Admins")
            Else
                Roles.AddUserToRole(txtAddUserName.Text.Trim, "Students")
```

```
            End If
            Membership.UpdateUser(user)
            txtAddUserName.Text = ""
            txtAddEmail.Text = ""
            txtAddFirstName.Text = ""
            txtAddLastName.Text = ""
            BindDataGrid()
        Else

            lblDBErrors.Text = status.ToString
        End If
    End Sub
End Class
```

## C.17 Edituserssubjects.ASPX

```
<%@ Page Language="vb" AutoEventWireup="false"
    MasterPageFile="~/default.Master" CodeBehind=
    "editusersubjects.aspx.vb" Inherits=
    "RemoteLab.editusersubjects" title="Untitled Page" %>
<asp:Content ID="Content1" ContentPlaceHolderID="cphMain" runat="server">
    <asp:ScriptManager ID="ScriptManager1" runat="server"></asp:ScriptManager>
    <asp:UpdatePanel ID="UpdatePanel1" runat="server">
        <ContentTemplate>
            <center>
                <h2>User in Subjects</h2>
                <asp:DropDownList ID="ddlUsers" runat="server"></asp:DropDownList>
                <asp:DropDownList ID="ddlSubjects" runat="server">
                    </asp:DropDownList>
                <asp:Label ID="lblUserName" runat="server" />
                <asp:DataGrid ID="dgdUserInSubject" runat="server"
                    HorizontalAlign="Center" CssClass="DGNormal"
```

```
                    AutoGenerateColumns="False" DataKeyField="ususername"
                    OnItemCommand="dgdUserInSubject_ItemCommand">
                    <Columns>
                    <asp:TemplateColumn HeaderStyle-Width="100px"
                        HeaderText="Subjects:" HeaderStyle-CssClass="DGHNormal">
                        <ItemTemplate>
                            <asp:Label ID="lblEditUsername" Text=
                                '<%#DataBinder.Eval(Container.DataItem,
                                "ussid")%>' Runat="server"/>
                        </ItemTemplate>
                    </asp:TemplateColumn>
                    <asp:ButtonColumn CommandName="Delete" Text="Delete" />
                </Columns>
            </asp:DataGrid>
            <asp:ValidationSummary ID="ValidationSummary2"
                DisplayMode="SingleParagraph" ShowSummary="True"
                ValidationGroup="Edit" Runat="server"
                HeaderText=
                "You must enter a valid input in the following fields:"/>
            <asp:Label ID="lblDBErrors" runat="server"
                CssClass="DBErrors"/>
            </center>
        </ContentTemplate>
    </asp:UpdatePanel>

</asp:Content>
```

## C.18 Edituserssubjects.ASPX.VB

```
Imports RemoteLab.MySQLConnection
Partial Public Class editusersubjects
    Inherits System.Web.UI.Page

    Protected Sub Page_Load(ByVal sender As Object,
        ByVal e As System.EventArgs) Handles Me.Load

    End Sub

    Protected Overrides Sub OnInit(ByVal e As System.EventArgs)
        If (Not IsPostBack) Then
            BindDataGrid()
        End If
    End Sub
    Private Sub BindDataGrid()
        Try
            Dim GroupsQuery As New MySQLQuery(QueryNums.SelectUsers,
                "ORDER BY comment")
            dgdUserInSubject.DataSource = GroupsQuery.Reader
            dgdUserInSubject.DataBind()
        Catch ex As Exception
        End Try
    End Sub
    Private Sub ClearDBErrors()
        lblDBErrors.Text = ""
    End Sub
    Private Sub ShowDBErrors(ByVal ex As MySQLException)
        lblDBErrors.Text = ex.message
    End Sub
    Sub dgdGroup_ItemCommand(ByVal sender As Object,
        ByVal e As DataGridCommandEventArgs)
```

```
        Select Case e.CommandName
            Case "Delete"
                Dim litGroupID As Literal = e.Item.FindControl("litEditGroupID")
                Try
                    Dim GroupsQuery As New MySQLQuery(QueryNums.DeleteGroup, "",
                        litGroupID.Text.Trim)
                Catch ex As MySQLException
                    ShowDBErrors(ex)
                End Try
                BindDataGrid()
            Case Else
        End Select
    End Sub

End Class
```

## C.19 Default.css

```
/* Page widths set for the body */
body
{
background-color:Gray;
font-family:Arial;
margin-left:auto;
margin-right:auto;
vertical-align:top;
width:990px;
}

table.tblMain
{
width:100%;
border-style:none;
}



table.tblMain td
{
text-align:left;
vertical-align:top;

}
h2, h3
{
padding:0px 10px 5px 10px;
}
h1{
text-align:center;
}
```

```
h2
{
text-decoration:underline;
}
h3
{
}

p, table {
padding:0px 20px 10px 20px;
}

.DBErrors
{
color:Red;
}



/* Default Setup for tables*/
table
{
border:outset thin InactiveBorder;
}
td, th
{
padding:3;
text-align:left;
}

/* The width for wide objects in the main place holder */
table.tblAddNewWide, .DGNormalWide, table.tblCenteredWide, table.tblWide
{
```

```
width:100%;
}

table.tblAddNew th, table.tblAddNewWide th, table.tblWide th,
    table.tblCenteredWide th
{
text-decoration:underline;
}
table.tblClear
{
border-style:none;
}

table.tblCenteredWide td
{
text-align:center;
}



/* Datagrid Settings */
.DGNormal, .DGNormalWide
{
background-color:Transparent;
font-size:small;
text-align:left;
font-weight:normal;
border:outset thin InactiveBorder;
}

.DGNormalWide{
width:95%
}
```

```
.DGNormal th, .DGNormalWide th
{

}

.DGNormal td, .DGNormalWide td
{
border-style:none;
padding:0px 4px 5px 4px;

}

/* Heading Cells for the DataGrids */
.DGHNormal, .DGHListOnly, .DGFListOnly
{
font-weight:bolder;
text-decoration:underline;
width:160px;
}

.DGHListOnly
{
}

.Number
{
text-align:right;
width:50px;
}
```

## C.20 Mysqlconnection.VB

```
Imports System.Data
Imports System.Data.Odbc
Namespace MySQLConnection
    Public Enum QueryNums
        SelectGroups = 1
        InsertGroup = 2
        AppendGroup = 3
        DeleteGroup = 4
        SelectSubjects = 5
        InsertSubject = 6
        AppendSubject = 7
        DeleteSubject = 8
        SelectUsers = 9
        AppendUser = 10
    End Enum
    Public Class MySQLQuery
        Private Shared ConnectionCount As Integer = 0
        Private P_Command As OdbcCommand
        Private P_Connection As New OdbcConnection
        Private P_DataReader As OdbcDataReader
        Private P_DataSet As DataSet
        Private Shared P_ConnectionString As String =
            "Driver={MySQL ODBC 3.51 Driver};Server=localhost;
            Database=rls;uid=InetUser;pwd=RLSUser;option=3"

        Private Function GetQueryStr(ByVal QRYNum As Integer)
            Select Case QRYNum
                Case 1
                    Return "SELECT * " & _
                           "FROM tblgroups"
                Case 2
```

```
            P_Command.Parameters.Add("@gname", OdbcType.VarChar, 255)
            Return "INSERT INTO tblgroups (gname) " & _
                    "VALUES (?)"
        Case 3
            P_Command.Parameters.Add("@gname", OdbcType.VarChar, 255)
            P_Command.Parameters.Add("@gid", OdbcType.Int)
            Return "UPDATE tblgroups " & _
                    "SET gname = ? " & _
                    "WHERE gid = ?"
        Case 4
            P_Command.Parameters.Add("@gname", OdbcType.VarChar, 255)
            Return "DELETE " & _
                    "FROM tblgroups " & _
                    "WHERE gid = ?"
        Case 5
            Return "SELECT * " & _
                    "FROM tblsubjects"
        Case 6
            P_Command.Parameters.Add("@sid", OdbcType.VarChar, 255)
            P_Command.Parameters.Add("@sname", OdbcType.VarChar, 255)
            Return "INSERT INTO tblsubjects (sid, sname) " & _
                    "VALUES (?, ?)"
        Case 7
            P_Command.Parameters.Add("@sname", OdbcType.VarChar, 255)
            P_Command.Parameters.Add("@sid", OdbcType.VarChar, 25)
            Return "UPDATE tblsubjects " & _
                    "SET sname = ? " & _
                    "WHERE sid = ?"
        Case 8
            P_Command.Parameters.Add("@sid", OdbcType.VarChar, 255)
            Return "DELETE " & _
                    "FROM tblsubjects " & _
                    "WHERE sid = ?"
```

```
            Case 9
                Return "SELECT PKID, Username, Comment, email, IsLockedOut "
                        "FROM mysql_membership"
            Case 10
                P_Command.Parameters.Add("@username", OdbcType.VarChar, 255)
                P_Command.Parameters.Add("@comment", OdbcType.VarChar, 255)
                P_Command.Parameters.Add("@email", OdbcType.VarChar, 255)
                P_Command.Parameters.Add("@IsLockedOut", OdbcType.VarChar)
                P_Command.Parameters.Add("@pkid", OdbcType.VarChar, 255)
                Return "UPDATE mysql_membership " & _
                        "SET username = ?, comment = ?, email= ?,
                        IsLockedOut = ? WHERE pkid = ?"
            Case Else
                Return ""
        End Select
    End Function
    Private Sub New(ByVal ExistingQuery As MySQLQuery)
        Me.P_Command = ExistingQuery.P_Command
        Me.P_Connection = ExistingQuery.P_Connection
        Me.P_DataReader = ExistingQuery.P_DataReader
        Me.P_DataSet = ExistingQuery.P_DataSet
    End Sub
    Public Sub New(ByVal Query As MySQLConnection.QueryNums,
        ByVal PostQueryString As String, ByVal ParamArray Parameters()
        As String)
        Dim x As Integer
        ' Trys to connect to the database and run the query
        Try
            P_Connection = New OdbcConnection(P_ConnectionString)
            P_Connection.Open()
            ConnectionCount = ConnectionCount + 1
            ' Connect to the database and run the query
            P_Command = New OdbcCommand
```

```
            ' Adds the post query to the end of the current query
            P_Command.CommandText = GetQueryStr(Query) & " " & PostQueryString
            ' If the query string was not found throw an error
            If P_Command.CommandText.Length < 1 Then
                Throw New MySQLException("Could not locate query string")
            End If
            ' Adds the connection to the command
            P_Command.Connection = P_Connection
            ' If the input Parameters and the required parameters are the same
            If Me.P_Command.Parameters.Count = Parameters.Length Then
                ' While there are more parameters in the parameter array add
                    value of the parameter to the command
                For x = 0 To (Parameters.Length - 1)
                    If Parameters(x) = "" Then
                        P_Command.Parameters(x).Value = DBNull.Value
                    ElseIf Parameters(x) = "True" Then
                        P_Command.Parameters(x).Value = 1
                    ElseIf Parameters(x) = "False" Then
                        P_Command.Parameters(x).Value = 0
                    Else
                        P_Command.Parameters(x).Value = Parameters(x)
                    End If
                Next
            Else
                Throw New MySQLException
                    ("Incorrect Number of Parameters Given")
            End If
            ' Saves the information into a reader
            P_DataReader = P_Command.ExecuteReader
                (CommandBehavior.CloseConnection)
            ' Converts the reader to a dataset
            'P_DataSet = convertDataReaderToDataSet(New MySQLQuery(Me).Reader)
        Catch ex As Odbc.OdbcException
```

```
            Throw New MySQLException(GetFriendlyErrorMessage(ex))
        End Try
    End Sub
    Public ReadOnly Property Reader() As OdbcDataReader
        Get
            Return New MySQLQuery(Me).P_DataReader
        End Get
    End Property
    'Public ReadOnly Property DataSet() As DataSet
    '    Get
    '        Return New MySQLQuery(Me).P_DataSet
    '    End Get
    'End Property
    Protected Overrides Sub finalize()
        Try
            P_Command.Dispose()
            P_DataReader.Close()
            P_DataSet.Clear()
            P_DataSet.Dispose()
        Catch ex As Exception
        Finally
            Try
                P_Connection.Close()
                P_Connection.Dispose()
                ConnectionCount = ConnectionCount - 1
            Catch ex As Exception
                Throw New MySQLException("Cannot disconnect from database")
            Finally
                MyBase.Finalize()
            End Try
        End Try
    End Sub
    Private Function GetFriendlyErrorMessage
```

```
        (ByVal ex As OdbcException) As String
        If ex.Message.Contains("Data too long for column") Then
            Return "One or more data fields is of the incorrect size."
        ElseIf ex.Message.Contains
            ("Cannot delete or update a parent row") Then
            Return "This record cannot be deleted as it is being used
                by other records."
        ElseIf ex.Message.Contains("Duplicate") Then
            Return "This record is already in database."
        ElseIf ex.Message.Contains("Data source name not found") Then
            Return "Could not connect to the database"
        ElseIf ex.Message.Contains("error in your SQL syntax") Then
            Return "Syntax Error"
        Else
            Return "An undefined error has occured"
        End If
    End Function
    Private Function convertDataReaderToDataSet
        (ByVal reader As OdbcDataReader) As DataSet
        Dim dataSet As DataSet = New DataSet()
        Dim schemaTable As DataTable = reader.GetSchemaTable()
        Dim dataTable As DataTable = New DataTable()
        Dim intCounter As Integer
        Try
            For intCounter = 0 To schemaTable.Rows.Count - 1
                Dim dataRow As DataRow = schemaTable.Rows(intCounter)
                Dim columnName As String = CType(dataRow("ColumnName"),
                    String)
                Dim column As DataColumn = New DataColumn(columnName,
                    CType(dataRow("DataType"), Type))
                dataTable.Columns.Add(column)
            Next
```

```
                dataSet.Tables.Add(dataTable)

                While reader.Read()
                    Dim dataRow As DataRow = dataTable.NewRow()
                    For intCounter = 0 To reader.FieldCount - 1
                        dataRow(intCounter) = reader.GetValue(intCounter)
                    Next
                    dataTable.Rows.Add(dataRow)
                End While
                Return dataSet
            Catch ex As Exception
                Return Nothing
            End Try
        End Function
    End Class
    Public Class MySQLException
        Inherits System.Exception
        Private P_Message As String
        Friend Sub New(ByVal Message As String)
            P_Message = Message
        End Sub
        Public Overrides ReadOnly Property message() As String
            Get
                Return P_Message
            End Get
        End Property
    End Class
End Namespace
```